TEKNILLINEN KORKEAKOULU

Sähkötekniikan osasto

Ilkka Hyytiäinen

DIGITAALIPIIRIEN TESTIVEKTOREIDEN AUTOMAATTINEN GENEROINTI

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 6.11.1989.

Työn valvoja Paavo Jääskeläinen

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä ja työn nimi:
Ilkka Hyytiäinen
Digitaalipiirien testivektoreiden automaattinen generointi
Päivämäärä: 06.11. 1989 **Sivumäärä:** 71

Osasto:
Sähköosasto

Professuuri:
Ele-66 Sovellettu Elektroniikka

Työn valvoja:
professori Paavo Jääskeläinen
Työn ohjaaja:

Tässä diplomityössä täydennettiin Micronas Oy:n suunnittelujärjestelmää digitaalikytkentöjen testivektoreiden generointijärjestelmällä. Toteutettu järjestelmä generoi testiherätteet lähes automaattisesti sekä synkroniselle että asynkroniselle digitaalikytkennälle. Kytkentä voi olla toteutettu joko ilman erityisiä testirakenteita tai käyttäen erityistä T-solurakennetta eli osittaista scan-rakennetta. Alkutietona käytetään kytkentäkaaviosta tuotettua kytkentälistaa.

Alkuosassa selvitettiin kirjallisuustyön tyyppisesti testingenerointiin liittyvää termistöä ja kuuden erilaisen algoritmin toiminta.

Seuraavaksi työssä esiteltiin yhdeksän kaupallista testingenerointijärjestelmää ja arvioitiin niiden soveltuvuutta ja liitettävyyttä Micronas Oy:n suunnittelujärjestelmään.

Kaupallisista järjestelmistä valittiin koekäyttöön kaksi soveltuvinta, jotka olivat HHB Systemsin Intelligen ja Zycad Corp:n Nextgen. Koekäytöt suoritettiin toimittajien tiloissa Lontoossa.

Tuloksien perusteella hankittavaksi valittiin Intelligen.

Käyttöönottovaiheesta tässä työssä on esitetty järjestelmän sijoittaminen Micronas Oy:n suunnittelujärjestelmään, ohjelmiston pääpiirteet ja siihen liittyvä T-solurakenne, työssä laaditut käyttörutiinit ja sovellusohjeet sekä työn osana kehitetty menetelmä T-solurakenteen hyödyntämiseksi käyttäen ainoastaan yhtä ylimääräistä digitaalituloa.

Lisäksi tässä työssä esitellään kaksi erilaista esimerkkitapausta testingeneroinnista.

Lopuksi esitetään joitakin suunnitteluohjeita, jotka helpottavat Intelligenin käyttöä logiikkasuunnittelussa.

Hakusanat : testivektori, scan-rakenne, Intelligen, testattavuus, vikasimulointi, kattavuus, algoritmi, logiikka, digitaali

ALKULAUSE

Tämä diplomityö on tehty Micronas Oy:n tuotekehitysosastolla. Micronas Oy:n innostunut ja uutta luova ilmapiiri toimi erinomaisena ympäristönä työn tekemiselle. Kiitän siitä koko henkilökuntaa.

Kiitän VTT-ELE:n Matti Weissenfeltia ja Pentti Paloa yhteistyöstä aineiston hankinnassa.

Tuotekehitysjohtaja Aarni Kajastetta kiitän mahdollisuudesta tehdä diplomityö hänen osastollaan.

Erityisesti haluan kiittää suunnittelupäällikkö DI Tapio Markkia, joka antoi kaiken mahdollisen apunsa aihepiiriin tutustumisessa sekä ongelmatilanteiden selvittämisessä.

Lisäksi haluan osoittaa erityiset kiitokset muille projektin puitteissa työskennelleille suunnittelijoille.

Lopuksi haluan kiittää vaimoani Raijaa antamastaan tuesta ja kannustuksesta diplomityön teossa.

Espoossa 6. marraskuuta 1989

SISÄLLYSLUETTELO

MERKINTÖJÄ, KÄSITTEITÄ JA LYHENTEITÄ

ASYNKRONINEN	asynkronisessa kytkennässä kellosignaalien aktiiviset muutokset eivät välttämättä tapahdu samanaikaisesti koko kytkennässä
ATPG	Automatic Test Pattern Generation
CLUSTER	Clusterin muodostavat tietokoneet voivat käyttää yhteisiä tietokantoja, niillä on yhteinen käyttöjärjestelmä ja systeemihallinta
CPU	Central Processing Unit
ETHERNET	erääntyyppinen nopea tiedonsiirtoyhteys tietokoneiden välillä
HAVAITTAVUUS	havaittavuus määrittelee työmäärän, joka tarvitaan kytkennän solmun tilan selvittämiseen
HERÄTE	kytkennän tulojen tilamäärittely
KATTAVUUS	kattavuus kertoo kuinka suuri osa kaikista mahdollisista vioista havaitaan
KOMBINAATIOELIN	piirielin, jonka lähtöjen tila on täysin riippuvainen kunkin hetkisestä tulojen tilasta
KYTKENTÄLISTA	kytkentälista määrittelee miten kytkennän piirielimet on kytketty toisiinsa
LOHKO	kytkentä muodostuu yleensä useasta kokonaisuuden muodostavasta lohkosta
MALLI, -KIRJASTO	simulaattoria varten tehdään piirielimien toimintaa kuvaavat mallit, jotka muodostavat mallikirjaston
MIPS	Million Instructions Per Second
OHJATTAVUUS	ohjattavuus määrittelee työmäärän, joka tarvitaan kytkennän solmun asettamiseksi tiettyyn tilaan
POLKU	signaalitie digitaalikytkennässä
PRIMÄÄRILÄHTÖ	lähtö, joka voidaan kytkeä piirin ulkopuoliseen järjestelmään
PRIMÄÄRITULO	tulo, johon voidaan kytkeä signaali piirin ulkopuolelta
PRIMITIIVI	simulaattorin sisäinen perusmalli loogiselle funktiolle
REDUNDANTTINEN	vika on redundanttinen, jos on olemassa toinen vika, jonka vaikutus piirin toimintaan on sama
S-A-1, S-A-0, STUCK-AT	vikamalli, jossa on oikosulku solmusta esim. käyttöjännitteeseen
SCAN	testimenetelmä, jossa kytkennän sisäiset solmut ovat ohjattavissa ja havaittavissa eräänlaisen siirtorekisteriketjun avulla
SEKVENSSIELIN	piirielin, joka vaatii vähintään kaksi ajassa toisiaan seuraavaa herätettä saavuttaakseen kaikki mahdolliset tilansa
SOLMU	vastaa pistettä, johon on kytketty signaaleja eri piirielimistä todellisessa kytkennässä
SYNKRONINEN	synkronisessa kytkennässä kaikki aktiiviset kellosignaalien muutokset tapahtuvat samanaikaisesti.

TESTATTAVUUS-ANALYYSI	menetelmä, jolla voidaan arvioida, kuinka helposti voidaan tehdä testiherätteet kytkennälle
TESTIKUVIO	testikuvio sisältää joukon testivektoreita
TESTIVEKTORI	testivektori sisältää testiherätteen ja sen-hetkisen vasteen
TILA	kytkennän solmulla on tila, joka voi olla joko looginen 1 tai looginen 0
TILAKONE	tilakone muodostuu sekvenssielinten ja kombinaatio-elinten yhdistelmästä jossa on usein mm. takaisin-kytkettyjä signaaleja
TOTUUSTAULU	totuustaulu kuvaa piirielimen toimintaa selvittäen mitkä ovat lähtöjen tilat tulojen tilojen funktiona
VASTE	kytkennän lähtöjen tilat kullakin hetkellä
VIKASIMULOINTI	vikasimulointi määrittelee niiden vikojen joukon, joiden vaikutus on nähtävissä primäärilähdöissä, kun kytkentää simuloidaan testiherätteillä

LIITTEET

1 JOHDANTO

1.1 YLEISTÄ

Asiakaspiirien digitaaliosien monimutkaistuessa on testiohjelmien laatimiseen kuluva aika muodostunut erittäin suureksi verrattuna muuhun suunnitteluun kuluvaan aikaan. Piirin suunnitteluun kuluvaa kokonaisaikaa ja myös kustannuksia voidaan merkittävästi pienentää tehokkaalla testingenerointityökalulla.

Micronas Oy:ssä tuotekehitysosaston erääksi avainalueeksi v. 1988 määriteltiin suunnittelutyön tehostaminen. Tavoitteena oli muunmuassa suunniteltavien piirien testien kehitysajan ja kehityskustannusten pienentäminen puoleen siten, että lopullinen tuotantotestausaika ei kasva nykyisestä. Nykyisen näkemyksen mukaan tavoitteeseen ei voida päästä kuin automatisoimalla testinkehitys. Koska piirien integrointiasteen ja koon kasvaessa saanto yleensä pienenee, on toimitusten laadun ylläpitämiseksi piirit testattava entistä kattavammalla testillä. Toisaalta, koska testin tekemiseen riittävän kattavaksi tarvittava työmäärä kasvaa eksponentiaalisesti suhteessa haluttuun kattavuuteen, on selvää, että testejä ei voida enää generoida manuaalisesti. On arvioitu, että tehokas testivektorigeneraattori pystyy pienentämään testin kehitysaikaa suurilla piireillä (n.4000 porttia) jopa neljännekseen, jonka voidaan arvioida vastaavan usean sadan tunnin työtä.

Markkinoilla on useita erityyppisiä automaattisia testivektorigeneraattoreita, mutta suurin osa niistä soveltuu joko pelkästään kombinaatiologiikan testin tekemiseen tai täydellisen scan-rakenteen avulla toteutetun logiikan testin tekemiseen.

Tässä diplomityössä on esitelty muutamia yleisimpiä testingenerointialgoritmeja ja kaupallisia ohjelmistoja. Varsinaiseen koeajovaiheeseen on otettu vain sellaiset generaattorit, jotka eivät välttämättä tarvitse scan-rakennetta sekventiaalisessa logiikassa generoidakseen testin ja jotka ovat muutenkin liitettävissä osaksi Micronas Oy:n suunnittelujärjestelmää. Koeajojen jälkeen suoritettiin valinta ja käyttöönotto, jossa luotiin käyttörutiinit ja sovellusohjeistot sekä menetelmä piilotetun osittaisen scan-rakenteen toteuttamiseksi. Lisäksi suoritettiin testingenerointi kahdelle erilaiselle esimerkkipiirille. Lopuksi esitettiin muutamia suunnitteluohjeita, joiden noudattaminen auttaa testingeneroinnissa.

1.2 DIPLOMITYÖN RAJAUKSESTA JA TOTEUTUKSESTA

Tämän diplomityön alkuosa koostuu kirjallisuusselvityksestä, jossa ensin esitellään yleisimmin käytetyt digitaalitestien generointialgoritmit. Seuraavaksi esitellään yleisimmät kaupalliset atpg-ohjelmistot ja arvioidaan niiden soveltuvuus Micronasin suunnittelujärjestelmän osaksi. Tämän jälkeen esitellään suoritetut koeajot, niiden yleiset järjestelyt, koeajoihin valitut koekytkennät, koeajoissa huomioitavat seikat ja tuloksien arviointi, sekä tehdään valinta. Käyttöönotosta tässä työssä esitellään ohjelmiston sijoittuminen suunnittelujärjestelmän osaksi, laaditut käyttörutiinit ja suunnitteluohjeistot ja -menetelmät. Lopuksi esitetään kaksi esimerkkitapausta ja joitakin suunnitteluohjeita sekä testingeneroinnin asettamat rajoitukset logiikan suunnittelulle.

Tämä diplomityö on tehty Micronas Oy:n palveluksessa tuotekehitysosastolla. Diplomityö kuuluu osana Menetelmäkehitys-88 -projektiin, jonka eräänä tavoitteena on automaattisten testivektorigeneraattoriohjelmien arviointi, järjestelmän valinta, hankinta ja käyttöönotto.

1.3 DIGITAALIPIIRIEN TESTAUKSESTA

Digitaalipiiri testataan loogisten toimintojen osalta syöttämällä piirille erilaisia digitaalisia herätteitä ja vertaamalla piirin lähdöissä olevia signaalitiloja odotettuihin tiloihin. Jos piirin käyttäytyminen on erilaista verrattuna odotettuun, on piiri viallinen. Testauksen käytännön järjestelyjä varten on olemassa erilaisia yleiskäyttöisiä tietokoneen ohjaamia testereitä, jotka pystyvät antamaan piirille suunnitellun herätteen ja samanaikaisesti vertaamaan piirin lähtöjen tiloja odotettuihin tiloihin. Herätteen vaatimuksena on, että mahdollisimman monien eri vikojen vaikutus on havaittavissa piirin lähdöissä eli saavutetaan mahdollisimman suuri kattavuus.

Herätteet ja niitä vastaavat vasteet on tuotettava jokaiselle piirityypille erikseen. Niiden tuottamista sanotaan testingeneroinniksi.

2 AUTOMAATTISET TESTIVEKTOREIDEN GENEROINTIALGORITMIT

2.1 KÄSITTEITÄ

2.1.1 Vikamallit

Tässä yhteydessä käsitellyt algoritmit käyttävät ns. 'stuck-at'-mallitusta , jossa vika on oikosulku solmusta käyttöjännitteeseen (s-a-1) tai maahan (s-a-0). Lisäksi erotellaan viat sekä portin tuloille että lähdöille, jolloin saadaan neljä eri vikatyyppiä. Ne ovat tulon oikosulku maahan, tulon oikosulku käyttöjännitteeseen, lähdön oikosulku maahan ja lähdön oikosulku käyttöjännitteeseen. Myöhemmin tässä työssä käytetään vian merkintöjä 's-a-0' ja 's-a-1'.

2.1.2 Vian herättäminen eli 'fault sensitizing' tai 'fault exercising'

Kun solmun vika on määritelty edellä mainitulla tavalla esimerkiksi oikosulkuna käyttöjännitteeseen, tarkoittaa vian herättäminen solmun tilan ohjaamista viattomassa tapauksessa tilaan 0 eli vian aiheuttaman tilan komplementtiin.

2.1.3 Vian eteneminen eli 'fault propagation'

Vian aiheuttama muutos vioitettavan solun tilassa verrattuna viattomaan tilaan on saatava etenemään vioitettavasta solmusta primäärilähtöön asti, jotta vika voidaan todeta. Vian paljastavan signaalin etenemistä piirissä kohti primäärilähtöä sanotaan vian etenemiseksi.

2.1.4 Vaatimusten täyttäminen eli 'justification'

Vian herättämisessä ja etenemisessä määritellään aina polku, jonka varrella olevien porttien muiden tulojen tilat asetetaan siten, että signaali kulkee porttien läpi. Piirin primäritulojen tilojen määrittämistä siten, että määritellyt tilat toteutuvat ilman ristiriitoja, sanotaan vaatimusten täyttämiseksi.

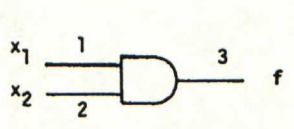
2.1.5 Käytetyt signaalitasoja osoittavat merkinnät

1	=	looginen 1
0	=	looginen 0
x	=	määrittelemätön tila
D	=	looginen 1 viattomassa piirissä, 0 viallisessa piirissä
/D	=	looginen 0 viattomassa piirissä, 1 viallisessa piirissä

2.1.6 Taulut

D-algoritmi ja sen kautta kaikki muut tässä esitetyt algoritmit käyttävät hyväkseen kolmea erityyppistä totuustaulua, jotka kuvaavat portin toimintaa.

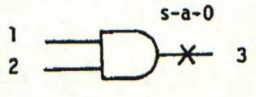
Supistettu totuustaulu eli 'singular cover' esittää tulojen vähimmäisvaatimukset portin lähtöjen kaikkien mahdollisten kombinaatioiden saavuttamiseksi. Supistetun totuustaulun yksittäistä riviä kutsutaan yksittäistauluksi ('singular cube').



1	2	3
1	1	1
0	X	0
X	0	0

Kuva 2.1 Supistettu totuustaulu. [1, s. 34]

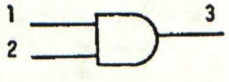
Testitaulu eli 'failure D-cube', joka on vikakohtainen, esittää ne tulojen tilat, jotka toteuttavat testin vialle eli ohjaavat portin lähdön vikatilannetta vastaavan tilan komplementtiin.



1	2	3
1	1	D

Kuva 2.2 Testitaulu. [1, s. 34]

Etenemistaulu eli 'propagation D-cube' esittää ne portin tulojen tilat, joilla johonkin portin tuloon ohjattu tila D tai /D etenee portin lähtöön.



AND

1	2	3	
0	0	0	α_1
1	0	0	α_2
0	1	0	α_3
1	1	1	α_4

	1	2	3
$\alpha_1 \cap \alpha_4$	$\overline{D}$	$\overline{D}$	$\overline{D}$
$\alpha_2 \cap \alpha_4$	1	$\overline{D}$	$\overline{D}$
$\alpha_3 \cap \alpha_4$	$\overline{D}$	1	$\overline{D}$
$\alpha_4 \cap \alpha_1$	D	D	D
$\alpha_4 \cap \alpha_2$	1	D	D
$\alpha_4 \cap \alpha_3$	D	1	D

Kuva 2.3 Täydellinen totuustaulu ja etenemistaulu.

Etenemistaulu saadaan johtamalla se täydellisestä totuustaulusta kuvassa 2.4 esitettyjen sääntöjen mukaan yhdistämällä pareittain eri tilan lähtöön tuottavat rivit.

$$0 \cap 0 = 0 \cap X = X \cap 0 = 0,$$

$$1 \cap 1 = 1 \cap X = X \cap 1 = 1,$$

$$X \cap X = X,$$

$$1 \cap 0 = D, \quad 0 \cap 1 = \bar{D}.$$

Kuva 2.4 Säännöt etenemistaulun tuottamiseen. [1, s. 35]

Esimerkiksi kuvan 2.3 etenemistaulun ensimmäinen rivi on saatu yhdistämällä säännön mukaisesti täydellisen totuustaulun ensimmäinen ja viimeinen rivi.

2.1.7 Peräytyminen eli 'back-tracking'

Kaikissa tässä esitetyissä algoritmeissa joudutaan tekemään enemmän tai vähemmän satunnaisia valintoja muunmuassa aseteltaessa solmujen tiloja vian herättämisen tai vian etenemisen yhteydessä. Usein väärän valinnan jälkeen joudutaan ristiriitatilanteeseen, jossa tietyn solmun tilan pitäisi olla toisaalta 1, toisaalta 0. Tällöin suoritetaan peräytyminen eli palataan takaisin sellaiseen vaiheeseen, jossa voidaan tehdä toinen valinta, ja palautetaan edellisen valinnan jälkeen tehdyt määrittelyt ennalleen.

2.1.8 D-rintama eli 'D-frontier'

Primäärilähtöä lähinnä olevaa porttia tai porttijoukkoa, jonka tulossa tai tuloissa on tila D tai /D, mutta lähdössä X, sanotaan D-rintamaksi.

2.1.9 Herkistetty polku eli 'sensitized path'

Sellainen porttiketju vioitettavasta solmusta primäärilähtöön, jonka signaalitien ulkopuoliset tulot on asetettu siten, että vian aiheuttama muutos aiheuttaa muutoksen myös lähdössä on nimeltään herkistetty polku.

2.1.10 Testi

Testi määrittelee tilat kytkennän primäärituloille. Tilojen määrittelyt ovat sellaiset, että vioitettava solmu ohjautuu viallisen tilan komplementtiin ja vioitettavan solmun tila etenee johonkin primäärilähtöön. Testivektori sisältää lisäksi herätettä vastaavan vasteen ehjälle piirille.

2.2 PERUSTAVOITE

2.2.1 Yleistä

Perustavoitteena on luoda ns. rakenteellinen testi eli testivektorit, jotka paljastavat stuck-at-malliset viat riittävän suurella kattavuudella. Rakenteellinen testi ei testaa piirin varsinaista toimintaa, vaan pyrkii löytämään vikamallituksen mukaiset yksittäiset viat. Toiminnallinen testi puolestaan selvittää, pystyykö piiri suoriutumaan sen varsinaisesta tehtävästä eli funktiosta.

2.2.2 Ongelmia

Käytännön ongelmana on piirien koon kasvaessa kytkennän syvyyden kasvu. Syvyyden kasvaessa on signaalipolulla yhä enemmän valintamahdollisuuksia. Valinnan virheellisyys paljastuu usein vasta, kun on tehty runsaasti laskentaa ja siten menetetty turhaan laskentatehoa. Väärien valintojen poistamiseksi tehtävät ratkaisut ovatkin osoittautuneet useimmin tehokkaimmiksi algoritmin kehittämismenetelmiksi.

2.2.3 Rajoitukset

Koska yleensä piirielinten mallituksessa testingenerointitarkoitukseen käytetään vain yksikköviiveitä, saattaa testin generointi olla epäluotettavaa esimerkiksi asynkroniselle logiikalle tai logiikalle, jossa on kriittisiä ajoituksia esimerkiksi kellosignaalien generoinnissa.

Koska tietokoneiden laskentakapasiteetti on rajallinen ja kytkentöjen koot ja vaikeusasteet usein hyvinkin vaativia, on useimmissa käytännön algoritmeissa rajattu toimintaa esimerkiksi rajoittamalla suurin sallittu peräytymiskertojen lukumäärä tai suurin sallittu yhden vian testin generointiin kuluva CPU-aika tiettyyn lukemaan.

2.3 JOHDANTO ALGORITMEIHIN

Kolmen viimeisen vuosikymmenen aikana on teollisuudessa ja erilaisissa tutkimuslaitoksissa kehitetty lukuisia erilaisia algoritmeja digitaalipiirien testivektoreiden generointiin. Eniten kehitystyötä on tehty USA:ssa ja Japanissa. Alkuun pyrittiin kehittämään menetelmiä kombinaatiologiikan testivektoreiden generointiin, jolloin syntyi useita erilaisia algoritmeja, jotka ovat kuitenkin melko paljon toistensa kaltaisia. Eräs ensimmäisistä toimivista algoritmeista oli ns. yksittäisen polun D-algoritmi, josta käytännössä lähes kaikki muut algoritmit polveutuvat.

Kombinaatiologiikan testingeneroinnin jälkeen huomio suunnattiin enemmän sekvenssilogiikan testingenerointiin. Tuloksena syntyi algoritmeja, jotka pystyvät generoimaan testin synkroniselle sekvenssilogiikalle tai jopa asynkroniselle sekvenssilogiikalle riippuen algoritmin tyypistä.

Seuraavassa esitellään joukko tunnettuja algoritmeja sekä sekvenssi- että kombinaatiologiikan testivektoreiden generointiin.

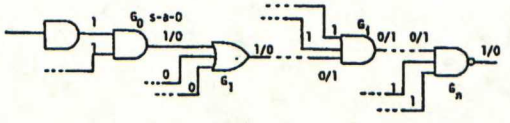
2.4 D-ALGORITMI

2.4.1 Yleistä

D-algoritmi on yksi vanhimmista testingenerointialgoritmeista, jonka perusmuodon kehitti Roth vuonna 1966. Sellaisenaan se kelpaa pääasiassa vain kombinaatiologiikan testingenerointiin, mutta siihen pohjautuen on kehitetty edelleen uusia algoritmeja erityisesti kombinaatiologiikan tarpeisiin.

2.4.2 Toimintaperiaate [1, s.30-45]

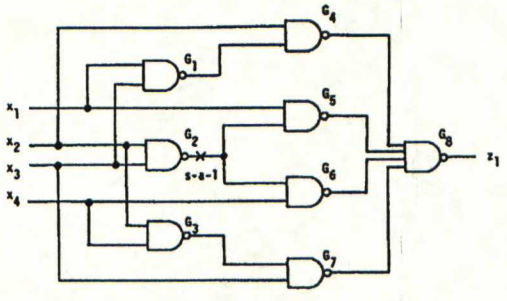
Yksinkertaisimmilaan D-algoritmi toimii ns. yksittäisen polun herkistämisen (one-dimensional path or single-path sensitization) periaatteella. Siinä määritellään ensin vikautettava solmu (G0:n lähtö) ja vian tyyppi (s-a-0) (kuva 2.5).



Kuva 2.5 Yksittäisen polun herkistäminen. [1, s. 31]

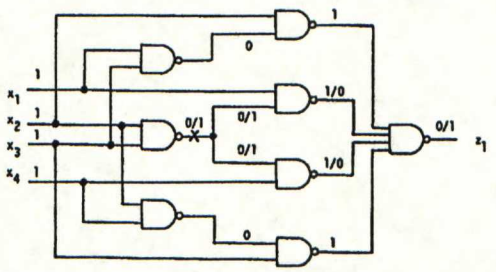
Tämän jälkeen määritellään G0:n tulot siten, että lähdön tila normaalisti olisi vikatilanteen tilan komplementti eli herätetään vika. Seuraavaksi määritellään polku vikautettavasta solmusta piirin primäärilähtöön G1 . . . Gn. Tätä vaihetta ja menetelmää kutsutaan nimillä 'forward-trace', 'error-propagation' tai 'D-drive' eli etenevä reititys. Syntynyt polku on herkistetty polku. Lopuksi on tarpeen löytää sellainen primääritulojen kombinaatio, joka toteuttaa määritellyt tilat polun porttien tuloissa. Tässä on kysymyksessä vaatimusten täyttäminen. Mikäli herkistettyjä polkuja muodostetaan vain yksi, saatetaan joissain tapauksessa ajautua tilanteeseen, jossa näennäisesti ei voida tuottaa testiä lainkaan.

Esimerkiksi kuvan 2.6 kytkennässä portin G2 lähdössä oleva vika ei yhden polun herkistämismenetelmällä näy koskaan piirin lähdössä. Yhden polun herkistämismenetelmässä etenemistaulut on johdettu supistetusta totuustaulusta.



Kuva 2.6 Tutkittava vika. [1, s. 32]

Jos poluksi valitaan vain G5-G8, pitäisi G6:n lähdön olla tilassa 1. Tämän saavuttamiseksi tulee x4:n olla tilassa 0. Siitä seurauksena G3:n lähtö on tilassa 1. Koska vian herättämiseen eli vioitettavan solmun ohjaamiseen vikatilanteen tilan komplementtiin, tässä tapauksessa tilaan 0, tarvitaan x3:een tila 1, syntyy G7:n lähtöön tila 0, joka estää vian näkymisen z1:ssä. Symmetrian vuoksi polku G6:n kautta kohtaa vastaavan ongelman. Jos kuitenkin herkistettyjä polkuja muodostettaisiin samanaikaisesti kaksi, sekä G5:n että G6:n kautta, saadaan vika näkymään z1:ssä kuvan 2.7 mukaisesti.



Kuva 2.7 Kahden polun samanaikainen herkistäminen. [1, s. 33]

Usean polun samanaikaiseen herkistämiseen perustuvaa menetelmää kutsutaan varsinaisesti D-algoritmiksi. Sen toteuttamiseksi käytetään hyväksi edellä esitettyjä totuustauluja sekä erityistä D-laskentaa. D-laskenta tarkoittaa käytännössä uuden testitaulun tuottamista vanhasta testitaulusta yhdistämällä se D-rintamalla olevan portin etenemistaulun kanssa tiettyjen sääntöjen (kuva 2.8) mukaan.

Coordinate D-intersection

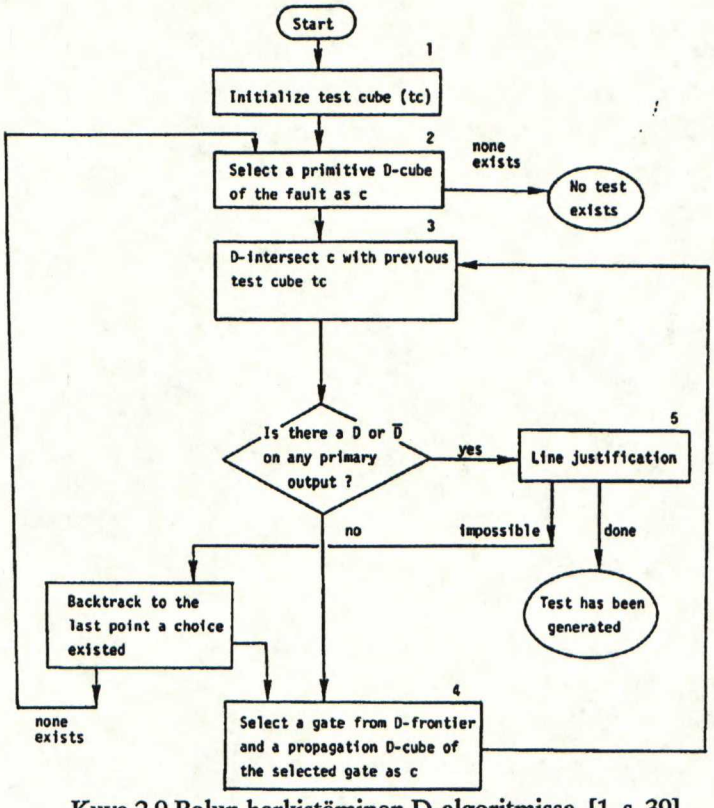
∩	0	1	x	D	$\bar{D}$
0	0	∅	0	ψ	ψ
1	∅	1	1	ψ	ψ
x	0	1	x	D	$\bar{D}$
D	ψ	ψ	D	D	ψ
$\bar{D}$	ψ	ψ	$\bar{D}$	ψ	$\bar{D}$

∅ = empty, ψ = undefined

Kuva 2.8 Säännöt testitaulun yhdistämiseksi etenemistaulun kanssa. [1, s. 37]

Algoritmin varsinainen toiminta jakaantuu kahteen eri vaiheeseen: polun herkistämiseen ja vaatimusten täyttämiseen. D-algoritmista on tehty useita eri versioita, jotka eroavat toisistaan mm. siinä, suoritetaanko vaatimusten täyttäminen aina, kun on määritelty polun varrella olevan portin jollekin tulolle arvo vai vasta, kun koko polku on määritelty vioitettavasta solmusta primäärilähtöön. Tässä esitetty algoritmi toimii jälkimmäisellä tavalla.

Polun herkistäminen tapahtuu kuvassa 2.9 esitetyn vuokaavion mukaisesti.

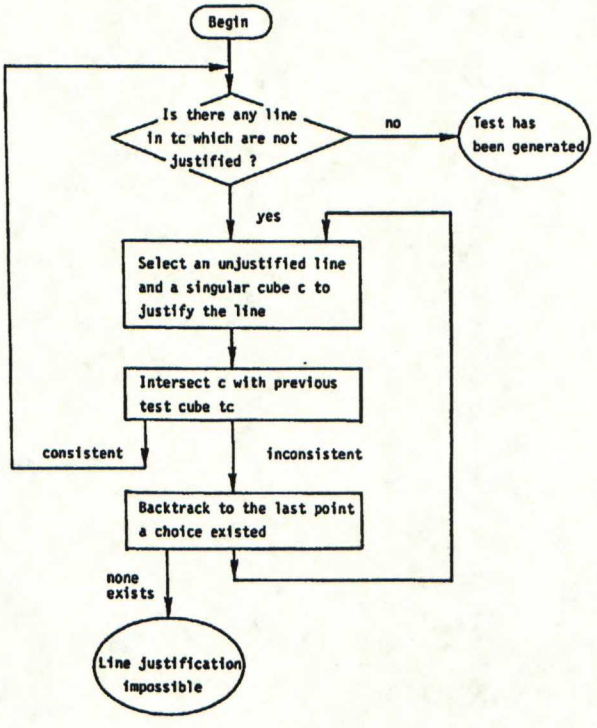


Kuva 2.9 Polun herkistäminen D-algoritmissa. [1, s. 39]

Alussa määritellään, mille vialle testi tehdään. Alustetaan ns. testitaulu eli varsinainen testivektoria vastaava piirin solmujen tilataulu eli määritellään kaikki solmut tuntemattomaan tilaan. Valitaan (satunnaisesti) vikataulusta yksi yksittäistaulu ko. vialle. Jos ko. vialle ei ole yksittäistaulua ko. portin vikataulussa tai jos kaikki mahdolliset valinnat on jo käytetty, ei testiä voi tehdä. Yhdistetään valittu yksittäistaulu edellisen testitaulun kanssa kuvassa 2.8 esitetyn taulukon mukaisesti. Tarkistetaan, onko jossain primäärilähdössä tila D tai /D. Jos on, siirrytään suorittamaan vaatimusten täyttämistä. Jos ei ole, valitaan (satunnaisesti) seuraava portti D-rintamalta.

Jos valintatilanne on sellainen, että kaikki vaihtoehdot on joko käytetty tai niitä ei ole, joudutaan palauttamaan tässä vaiheessa tehdyt määrittelyt ennalleen ja palaamaan takaisin edelliseen valintatilanteeseen.

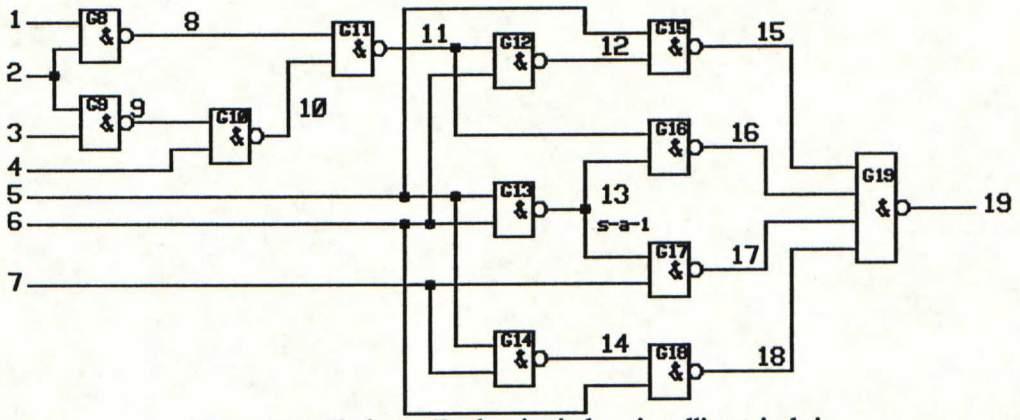
Polun herkistämisen jälkeen on huolehdittava siitä, että polun varrella olevat solmut saadaan määritettyihin tiloihin. Tätä vaihetta sanotaan vaatimusten täyttämiseksi, ja se tapahtuu kuvassa 2.10 esitetyn vuokaavion mukaisesti.



Kuva 2.10 Vaatimusten täyttäminen. [1, s. 43]

Tutkitaan, onko tuotetussa testitaulussa yhtään solmua, jonka tilan vaatimuksia sitä ohjaavien porttien osalta ei ole täytetty. Valitaan yksi solmu ja sitä ohjaavan portin yksittäistaulu, joka yhdistetään tuotetun testitaulun kanssa. Tarkistetaan, syntyikö ristiriitoja edellisten määrittelyjen kanssa. Jos ristiriitoja syntyi, on peräydyttävä takaisin sellaiseen vaiheeseen, jossa tehtiin valinta ja poistettava kaikki ko. valinnan jälkeen tehdyt määrittelyt. Jos ristiriitoja ei esiintynyt, siirrytään alkuun etsimään seuraavaa solmua, jonka tilan asettamisvaatimuksia ei ole täytetty. Kun kaikkien solmujen tilojen asettamat vaatimukset on täytetty eikä ristiriitoja ole voimassa, on testi generoitu. Varsinaisen testivektorin muodostavat ne testitaulun signaalit, jotka ovat kytkennän primäärituloja tai primäärilähtöjä.

2.4.3 Esimerkki D-algoritmin toiminnasta.



Kuva 2.11 Kytkentä D-algoritmin havainnollistamiseksi

Alustetaan koko kytkennän testitaulu.

```
Solmu :1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16171819
Tila :  x x x x x x x x x x  x  x  x  x  x  x x x x
```

Aloitetaan vian eteneminen. Yhdistetään koko kytkennän vikatauluun valitun vian testitaulusta yksi satunnainen halutun tilan toteuttava rivi. Tässä tapauksessa muita vaihtoehtoja ei tosin olisi.

```
tt0     x x x x x x x x x x x x x x x x x x x
                  1 1                /D
tt1     x x x x 1 1 x x x x x x /D x x x x x x
```

Valitaan D-rintamalta (G16 ja G17) G16, ja yhdistetään sen etenemistaulu nykyiseen testitauluun.

```
tt1     x x x x 1 1 x x x x x x /D x x x x x x
                              1  /D    D
tt2     x x x x 1 1 x x x x 1 x /D x x D x x x
```

Valitaan D-rintamalta (G17 ja G19) G19, ja yhdistetään sen etenemistaulun satunnaisesti valittu rivi '1 D 1 1 /D' nykyiseen testitauluun.

```
tt2     x x x x 1 1 x x x x 1 x /D x x D x x x
                                    1 D 1 1 /D
tt3     x x x x 1 1 x x x x 1 x /D x 1 D 1 1 /D
```

Koska tila /D on nyt saavuttanut primäärilähdön, siirrytään vaatimusten täyttämiseen.

Täytetään solmun 15 vaatimukset. Koska portin G15 lähdön pitää olla tilassa 1 ja solmu 5 on jo määritelty tilaan 1, valitaan portin G15 yksikkötauluista ' x 0 1' ja yhdistetään se nykyiseen testitauluun.

```
tt3    x x x x 1 1 x x x x 1 x /D x 1 D 1 1 /D
                 x               0     1
tt4    x x x x 1 1 x x x x 1 0 /D x 1 D 1 1 /D
```

Täytetään solmun 17 vaatimukset. Portin G17 lähdön pitää olla tilassa 1. Koska solmu 13 on jo määritelty tilaan /D, yhdistetään seuraavasti:

```
tt4    x x x x 1 1 x x x x 1 0 /D x 1 D 1 1 /D
                   0             x         1
tt5    x x x x 1 1 0 x x x 1 0 /D x 1 D 1 1 /D
```

Täytetään solmun 18 vaatimukset. Portin G18 lähdön pitää olla tilassa 1. Siihen päästään, jos jompikumpi tai molemmat solmuista 14 ja 6 on tilassa 0.

```
tt5    x x x x 1 1 0 x x x 1 0 /D x 1 D 1 1 /D
                 x                 0     1
tt6    x x x x 1 1 0 x x x 1 0 /D 0 1 D 1 1 /D
```

Täytetään solmun 12 vaatimukset. Portin G12 lähdön pitää olla tilassa 0 eli molemmat tulot tilassa 1.

```
tt6    x x x x 1 1 0 x x x 1 0 /D 0 1 D 1 1 /D
                 1           1 0
tt7    x x x x 1 1 0 x x x 1 0 /D 0 1 D 1 1 /D
```

Täytetään solmun 14 vaatimukset. Portin G14 lähdön pitää olla tilassa 0 eli molemmat tulot tilassa 1.

```
tt7    x x x x 1 1 0 x x x 1 0 /D 0 1 D 1 1 /D
                 1   1             0
tt8    x x x x 1 1 ? x x x 1 0 /D 0 1 D 1 1 /D
```

Vaatimusten täyttäminen epäonnistui, koska solmulle 7 oli aiemmin määritelty tila 0 ja nyt tilaksi halutaan 1. Tässä tapauksessa joudutaan perääntymään takaisin sellaiseen vaiheeseen, jossa on mahdollisuus tehdä jokin toinen valinta. Palautetaan takaisin ennen valintaa vallinnut testitaulu.

```
tt2    x x x x 1 1 x x x x 1 x /D x x D x x x
```

Valitaan portin G19 etenemistaulusta rivi '1 D D 1 /D' ja yhdistetään se nykyiseen testitauluun.

```
tt2    x x x x 1 1 x x x x 1 x /D x x D x x x
                                    1 D D 1 /D
tt3    x x x x 1 1 x x x x 1 x /D x 1 D D 1 /D
```

Koska tila /D on saavuttanut primäärilähdön, voidaan taas ryhtyä vaatimusten täyttämiseen. Täytetään solmun 15 vaatimukset eli määritellään tulot siten, että lähtö on tilassa 1.

```
tt3    x x x x 1 1 x x x x 1 x /D x 1 D D 1 /D
               x             0     1
tt4    x x x x 1 1 x x x x 1 0 /D x 1 D D 1 /D
```

Täytetään solmun 17 vaatimukset.

```
tt4    x x x x 1 1 x x x x 1 0 /D x 1 D D 1 /D
                 1           /D       D
tt5    x x x x 1 1 1 x x x 1 0 /D x 1 D D 1 /D
```

Täytetään solmun 18 vaatimukset.

```
tt5    x x x x 1 1 1 x x x 1 0 /D x 1 D D 1 /D
                 x                 0       1
tt6    x x x x 1 1 1 x x x 1 0 /D 0 1 D D 1 /D
```

Täytetään solmun 12 vaatimukset.

```
tt6    x x x x 1 1 1 x x x 1 0 /D 0 1 D D 1 /D
                 1           1 0
tt7    x x x x 1 1 1 x x x 1 0 /D 0 1 D D 1 /D
```

Täytetään solmun 14 vaatimukset.

```
tt7    x x x x 1 1 1 x x x 1 0 /D 0 1 D D 1 /D
               1   1               0
tt8    x x x x 1 1 1 x x x 1 0 /D 0 1 D D 1 /D
```

Täytetään solmun 11 vaatimukset.

```
tt8    x x x x 1 1 1 x x x 1 0 /D 0 1 D D 1 /D
                         0   x 1
tt9    x x x x 1 1 1 0 x x 1 0 /D 0 1 D D 1 /D
```

Täytetään solmun 8 vaatimukset.

```
tt9    x x x x 1 1 1 0 x x 1 0 /D 0 1 D D 1 /D
       1 1             0
tt10   1 1 x x 1 1 1 0 x x 1 0 /D 0 1 D D 1 /D
```

Nyt voidaan todeta, että enää ei ole jäljellä portteja, joiden lähdön tilaa ei ole tulojen osalta määritelty, joten testi on generoitu. Tilaltaan määrittelemättömät primääritulot tilat asetetaan esimerkiksi tilaan 0.

2. 4. 4 Ominaisuuksia

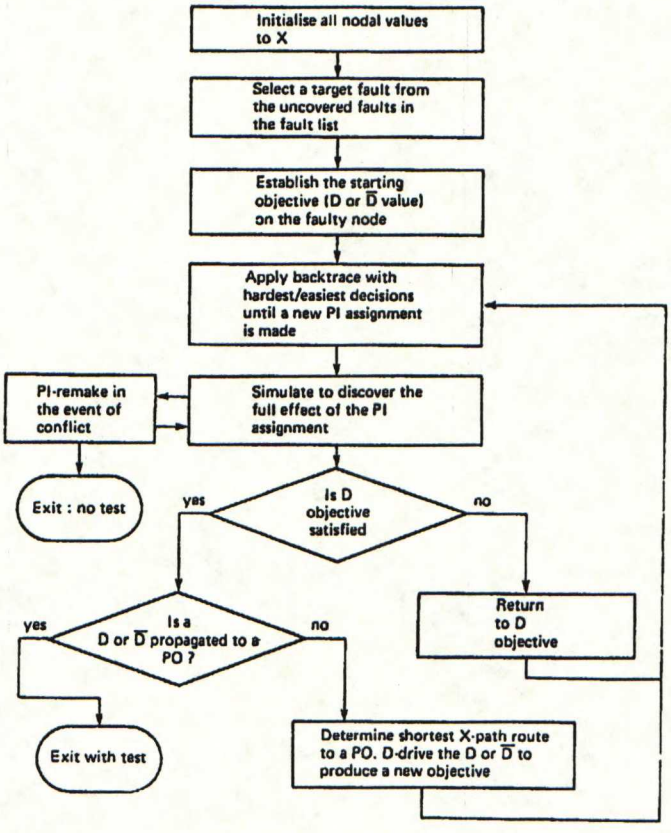
D-algoritmi vaatii paljon laskentaa. Koska vaatimusten täyttämisessä tehdään satunnaisia valintoja signaalireitin valitsemiseksi, on todennäköistä, että melko usein tehdään väärä valinta ja siten joudutaan usein peräytymään. D-algoritmin on todettu olevan tehoton piirille, joka sisältää runsaasti exor-rakenteita ja uudelleen yhtyviä signaaliteitä esimerkiksi virheenkorjaavissa rakenteissa.

2.5 PODEM

2.5.1 Yleistä

Podem-algoritmin (Path Oriented DEcision Making) kehitti vuonna 1981 P. Goel tarkoituksenaan luoda algoritmi, joka on parempi kuin D-algoritmi. D-algoritmin heikkoutena on erityisesti tilojen määrittäminen piirin sisäisille solmuille. Koska yleensä solmun tilaa määritellessä tai porttia (eli samalla solmua) valittaessa joudutaan tekemään satunnainen valinta, tehdään satunnainen valinta vähintään yhtä usein kuin piirissä on portteja. Tästä on seuraksena runsaat peräytymiset sekä niitä ennen tehty runsas ylimääräinen työ. Podem-algoritmissa valintoja tehdään etupäässä vain asetettaessa tiloja primäärituloihin, joten jo pelkästään mahdollisuuksia väärän valinnan tekemiseen on huomattavasti vähemmän. Lisäksi Podem-algoritmi pyrkii optimoimaan tehtyjä valintoja siten, että turhan laskennan tekemisen todennäköisyys on mahdollisimman pieni.

2.5.2 Toimintaperiaate [2, s.89-98]



Kuva 2.12 Podem-algoritmin vuokaavio. [2, s.99]

Podem-algoritmissa aloitetaan initialisoimalla kaikki kytkennän solmut määrittelemättömään tilaan. Tämän jälkeen valitaan vika, jolle ei ole vielä tehty testiä eli ns. kohdesolmuksi se solmu, jossa vika on, jolloin ns. tavoitetilaksi kohdesolmulle saadaan vikatilannetta vastaavan tilan komplementti.

Kirjallisuudessa kohdesolmulle ja tavoitetilalle käytetään esitystä (v,s), jossa v on tavoitetila ja s on solmun nimi.

Koko algoritmin aikana talletetaan aina valintatilanteessa muut jäljellä olevat valintavaihtoehdot mahdollisen perääntymisen varalta siten, että osataan palata samaan vaiheeseen, tehdä uusi valinta ja jatkaa uudelleen eteenpäin.

Seuraavaksi aloitetaan 'backtrace' eli taaksepäin suuntautuva jäljitys. Siinä aloitetaan aina kohdesolmusta ja edetään takaperin portti portilta kohti primäärituloja siten, että aina, jos reitin varrella olevan portin jokin tulo on määrittelemättömässä tilassa, määritellään sille oikea tila. Kuitenkin yhdellä läpikäynnillä kohdesolmusta primärituloon määritellään vain enintään yksi tulo kullekin portille. Kun on saavutettu primääritulo, suoritetaan logiikkasimulointi, jonka tarkoituksena on tarkistaa, onko haluttu tila saavutettu kohdesolmussa ja toisaalta saada uudet lähtötilat kytkennän solmuille uutta läpikäyntiä varten.

Jäljityksessä solmujen tiloja asetettaessa käytetään D-algoritmista tuttuja totuustauluja ja noudatetaan seuraavaa ohjetta: Jos portin lähdön asettamiseen tiettyyn tilaan tarvitaan vain yhden tulon asettaminen (esim. JA-portin lähtö on tilassa 0, jos vähintään toinen tuloista on tilassa 0), valitaan sellainen tulo, jonka ohjaus on helpointa. Jos portin kaikkien tulojen asettaminen on välttämätöntä, ryhdytään ensinnä asettamaan sellaista tuloa, jonka ohjaaminen on vaikeinta. Jos yhtä helposti ohjattavia solmuja on useita, valitaan niistä jokin satunnaisesti. Solmun ohjaamisen helppoudesta kertoo esimerkiksi jokin ohjattavuusanalyysi, joka laskee kullekin solmulle oman ohjattavuuslukunsa. Podem-algoritmin yhteydessä on käytetty muunmuassa Camelot-nimistä algoritmia.

Jos haluttua tilaa ei ole saavutettu, otetaan lähtötilanteeksi simuloinnissa saadut solmujen tilat, säilytetään kohdesolmu ja tavoitetila ennallaan ja aloitetaan taaksepäin suuntautuva jäljitys uudelleen. Jos haluttu tila saatiin aikaiseksi, siirrytään algoritmin seuraavaan vaiheeseen eli tarkistetaan, onko vikasignaali eli D tai /D edennyt primärilähtöön. Jos ei ole, siirrytään 'D-drive'-vaiheeseen eli D-rintaman siirtoon.

Jos simulointi osoittaa, että on syntynyt ristiriita eli kohdesolmu on joutunut tavoitetilan komplementtiin, ryhdytään 'PI-remake'-rutiiniin eli primäärituloj en uudelleen määrittelyyn, jossa viimeiseksi määritellyn primääritulon tila komplementoidaan, suoritetaan simulointi ja tarkistetaan vaikutus. Mikäli muutos ei poista ristiriitaa, komplementoidaan edelliseksi määritellyn primääritulon tila, simuloidaan ja tarkastetaan. Näin jatketaan, kunnes ristiriita on poistunut tai kaikki määriteltyjen primäärituloj en tilojen kombinaatiot on käyty läpi, jolloin voidaan todeta, että testiä ei voi tehdä.

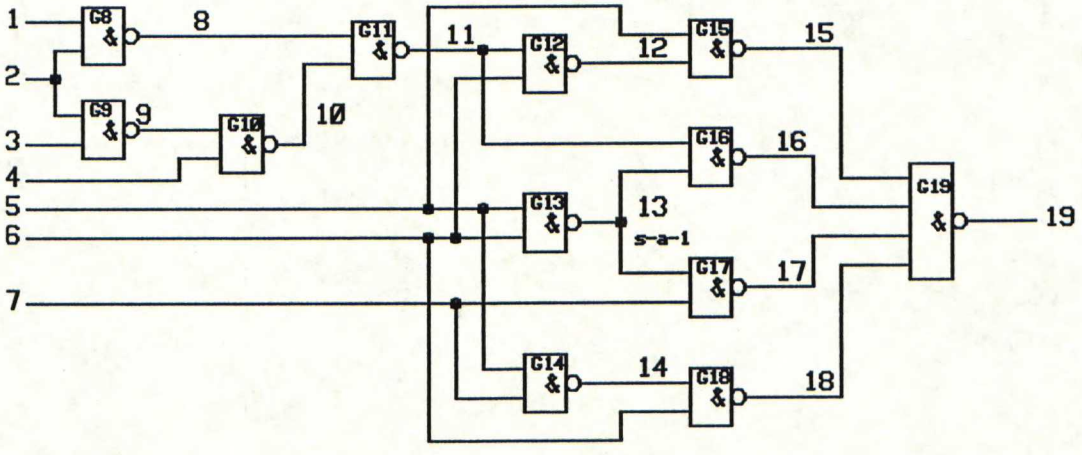
D-rintaman siirrossa ensin suoritetaan määrittelemättömän polun olemassaolon tarkistus eli tutkitaan onko olemassa sellaista polkua mistään D-rintaman

portista, jonka varrella kaikkien porttien lähdöt ovat määrittelemättömässä tilassa. Jos sellaista polkua ei ole, joudutaan peräytymään. Mikäli määrittelemättömiä polkuja löytyi, valintaan niistä se, joka on porteissa mitattuna lähinnä primäärilähtöä ja vaihdetaan uudeksi kohdesolmuksi ensimmäinen sen polun portti. Kohdesolmun tavoitetila saadaan D-algoritmin etenemistaulusta.

Uuden kohdesolmun ja tavoitearvon määrittelyn jälkeen aloitetaan taaksepäin suuntautuva jäljitys.

Näin jatketaan, kunnes testi on tehty tai on todettu, ettei testiä voi tehdä.

2.5.3 Esimerkki



Kuva 2.13 Kytkentä Podem-algoritmin havainnollistamiseksi.

Valitaan viaksi kuvan 2.13 kytkennän solmun 13 oikosulku käyttöjännitteeseen eli 13 s-a-1.

Kytkennän solmuille on Camelot-tyyppisellä ohjattavuusanalyysillä laskettu seuraava taulukko:

Solmu	Ohjattavuus
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	0.5
9	0.5
10	0.375
11	0.219

12	0.305
13	0.5
14	0.5
15	0.326
16	0.180
17	0.375
18	0.375
19	0.079

Taulukko 2.1 Esimerkkikytkennän solmujen ohjattavuus Camelot-algoritmin mukaan.

Solmu	Aloitus	Määr.	Simul.	Määr.	Simul.	Määr.	Simul.	Määr.	Simul.	Määr.	Simul.	Määr.	Simul.
1:	x		x		x	1	1	1	1	1	1	1	1
2:	x		x		x		x	1	1	1	1	1	1
3:	x		x		x		x		x		x		x
4:	x		x		x		x		x		x		x
5:	x	1	1	1	1	1	1	1	1	1	1	1	1
6:	x		x	1	1	1	1	1	1	1	1	1	1
7:	x		x		x		x		x	0	0	1	1
8:	x		x		x	0	x	0	0	0	0	0	0
9:	x		x		x		x		x		x		x
10:	x		x		x		x		x		x		x
11:	x		x		x	1	x	1	1	1	1	1	1
12:	x		x		x		x		0	0	0	0	0
13:	x	/D	x	/D	/D	/D	/D	/D	/D	/D	/D	/D	/D
14:	x		x		x		x		x		1		0
15:	x		x		x		x		1	1	1	1	1
16:	x		x		x	D	x	D	D	D	D	D	D
17:	x		x		x		x		x	1	1	1	D
18:	x		x		x		x		x		0		1
19:	x		x		x		x		x	/D	1	/D	/D

Taulukko 2.2 Podem-algoritmin eteneminen esimerkkikytkennässä.

Taulukossa 2.2 on esitetty Podem-algoritmin eteneminen kuvan 2.13 kytkennässä.

Alkuun kaikki kytkennän solmut alustetaan määrittelemättömään tilaan.

Sen jälkeen määritellään vioitettava solmu ja vian tyyppi eli kohdesolmuksi 13 ja vikatyypiksi s-a-1, eli kohdesolmun 13 tavoitetila on /D eli 0.

Määritetään portin 13 vaikeimmin ohjattava tulo eli solmulle 5 tila 1.

Suoritetaan simulointi ja todetaan, että haluttua vaikutusta vioitettavassa solmussa ei saatu aikaiseksi.

Määritetään portin 13 vaikeimmin ohjattava määrittelemätön tulo eli solmulle 6 tila 1. Suoritetaan simulointi ja todetaan, että kohdesolmu on halutussa tilassa eli solmu 13 on tilassa 0. Siirrytään D-rintaman siirtoon.

Vanhasta D-rintamasta on kaksi määrittelemätöntä polkua primäärilähtöön eli polku G16:n kautta ja polku G17:n kautta. Valitaan satunnaisesti G16, jonka lähdöstä tehdään uusi kohdesolmu. Tavoitetilaksi määritellään D eli 1.

Määritellään etenemistaulu ja testitaulu yhdistämällä solmulle 11 tila 1, portin G11 helpoimmin ohjattavalle tulolle eli solmulle 8 tila 0 ja portin G8 tulolle solmulle 1 tila 1. Suoritetaan simulointi ja todetaan, ettei kohdesolmu saanut haluttua tilaa. Suoritetaan läpikäynti uudelleen ja tuloksena määritellään solmulle 2 tila 1.

Suoritetaan simulointi ja todetaan, että kohdesolmu 16 on saanut halutun tilan.

Siirretään D-rintamaa eteenpäin ja saadaan uudeksi kohdesolmuksi 19 ja tavoitetilaksi /D eli 0.

Määritellään portin 19 vaikeimmin ohjattava määrittelemätön tulo etenemistaulun mukaisesti tilaan 1. Koska sekä solmu 17 että 18 ovat yhtä helposti ohjattavissa, valitaan satunnaisesti solmu 17, jolle siis määritellään tila 1. Sen seurauksena solmulle 7, joka onkin primääritulo, määritellään tila 0.

Suoritetaan simulointi ja todetaan, että solmun 19 tavoitetilan /D sijaan se onkin tilassa 1 eli on syntynyt ristiriita. Peräydytään edelliseen päätöksentekovaiheeseen eli päätökseen antaa solmulle 7 tila 0. Vaihdetaan solmun 7 tilaksi 1 ja suoritetaan simulointi. Todetaan, että kohdesolmu 19 on saanut tavoitetilan /D eli 0. Koska /D (tai D) on nyt edennyt primäärilähtöön, on testi tehty.

Määrittelemättömät primääritulot asetetaan esimerkiksi tilaan 0.

2.5.4 Ominaisuuksia

Podem-algoritmi on selvästi tehokkaampi kuin D-algoritmi pienemmän peräänty mismäärän vuoksi. Perääntymistarvetta vähentää selvästi ohjattavuusanalyysin antamien tietojen hyväksikäyttö signaaliteitä valittaessa. Kuitenkin Podem tekee vääriä valintoja ja sen vuoksi laskentatehoa käytetään turhaan.Vääriä valintoja voidaan välttää tarkemmalla ja perustellummalla valinnalla. Tähän asiaan on perehdytty tarkemmin Fan-algoritmissa, joka esitellään seuraavaksi.

2.6 FAN

2.6.1 Yleistä

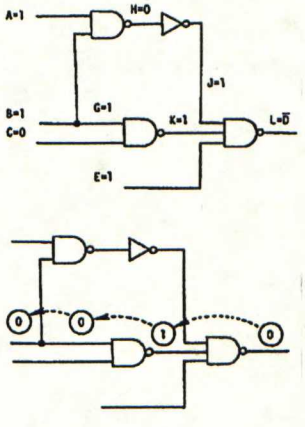
Fan-algoritmin kehittivät Podem-algoritmista Fujiwara ja Shimono. He huomasivat Podem-algoritmin suurimman ongelman olevan edelleen liian suuren hukkaanmenneen työmäärän, jos joudutaan perääntymään. Fan-algoritmissa on useita erilaisia Podem-algoritmista puuttuvia strategioita, joiden tarkoituksena on turhien perääntymisten estäminen. Niiden ansiosta Fan-algoritmi on 1 - 6 kertaa tehokkaampi kuin Podem

2.6.2 Toimintaperiaate [1, s.53-67]

Perustoiminnot, kuten esimerkiksi etenevä reititys ja taaksepäin suuntautuva reititys, ovat Fan-algoritmissa samanlaiset kuin Podem-algoritmissa. Perustoimintojen käyttöä kuitenkin ohjataan kuudella erityisohjeella, joita kirjallisuudessa kutsutaan strategioiksi.

Ohje 1: Yleisesti jokaisessa algoritmin vaiheessa määritellään mahdollisimman monet sellaiset solmut tiettyyn tunnettuun tilaan, jotka vaikuttavat testin olemassaoloon. Tässä auttaa 'backward implication', eli eräänlainen ajassa takaperin toimiva simulaattori, joka määrittelee kytkennän tuloille tilat riippuen jonkin solmun tilasta.

Ohje 2: Määritettäessä vikautettavalle solmulle tilaa D tai /D, suoritetaan määrittely ohjeen 1 mukaisesti eli asetetaan kaikki tarpeelliset primääritulot tiettyihin tiloihin. Kuvan 2.14 yläosassa nähdään tilanne Fan-algoritmin tilojen määrittelyn jälkeen, kun taas kuvan alaosassa nähdään tilanne Podem-algoritmin taaksepäin suuntautuvan jäljityksen jälkeen. Kuten voidaan havaita, on Podem tehnyt virheen asettaessaan solmun B tilaan 0, joten peräytyminen on odotettavissa, kun taas Fan pystyi tuottamaan testin suoraan.



Kuva 2.14 Ohjeen 1 ja 2 antama etu Podem-algoritmiin nähden. [1, s.55]

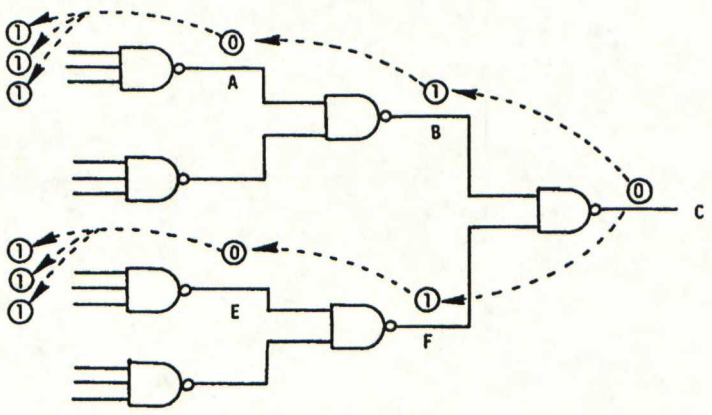
Ohje 3: Mikäli kaikki polut vikautettavasta solmusta primäärilähtöihin kulkevat jossain kohdassa tai kohdissa yhden kaikille poluille yhteisen portin tai porttien kautta, on suoritettava polun osittainen herkistys, joka tehdään täydellisenä siten, että kaikki primääritulot, jotka vaikuttavat yhteisten porttien herkistymiseen, asetetaan pysyvästi tiettyyn tilaan.

Ohje 4: Tähän ohjeeseen liittyvät uudet käsitteet ovat 'free' eli vapaa, 'bound' eli sidottu ja 'head' eli ohjaava solmu. Jos solmuun S on olemassa polku jostain sellaisesta solmusta, joka ohjaa suoraan vähintään kahta eri porttia, on S sidottu solmu. Jos solmu ei ole sidottu, se on vapaa. Vapaa solmu, joka ohjaa suoraan sidottua solmua, on ohjaava solmu.

Keskeytetään taaksepäin suuntautuva jäljitys ohjaavaan solmuun, ja jätetään ohjaavan solmun osalta vaatimusten täyttäminen tehtäväksi myöhemmin.

Jos taaksepäin suuntautuva jäljitys tehdään vain enintään ohjaavaan solmuun asti, jää kytkennästä jäljelle vain sellainen osa, jossa ei ole solmuja, jotka ohjaisivat välittömästi kahta tai useampaa tuloa. Näin ollen myöhemmin tehtävä vaatimusten täyttäminen onnistuu varmasti ilman perääntymisiä.

Ohje 5: Suoritetaan taaksepäin suuntautuva jäljitys useammalle kuin yhdelle polulle samanaikaisesti esimerkiksi kuvan 2.15 mukaisesti eli suoritetaan monipolkuinen taaksepäin suuntautuva jäljitys.



Kuva 2.15 Monipolkuinen taaksepäin suuntautuva jäljitys. [1, s.59]

Kuvan 2.15 tapauksessa Podem suorittaa taaksepäin suuntautuvan jäljityksen kolmesti polulla C-B-A ja samoin kolmesti polulla C-F-E, ennenkuin tavoitetila 0 on saavutettu solmussa C.

Fan-algoritmin monipolkuisen taaksepäin suuntautuvan jäljityksen algoritmi on seuraavanlainen:

Algoritmissa määritellään kohde merkinnällä (s,n0(s),n1(s)), jossa s tarkoittaa kohdesolmua, n0(s) ilmoittaa luvun, kuinka monessa tilanteessa kohdesolmulle s annettiin tila 0 ja n1(s) ilmoittaa luvun, kuinka monessa tilanteessa kohdesol-

mulle s annettiin tila 1. Eteneminen taaksepäin aloitetaan alkukohteiden joukosta. Kulloinkin työn alla oleva solmu on nimeltään aktiivinen kohde. Ohjaavat solmut muodostavat ohjaavien kohteiden joukon ja vastaavasti ne solmut, jotka haarautuvat suoraan kahteen tai useampaan eri solmuun ('fanout-point'), muodostavat haarautumiskohteiden joukon.

Kuljettaessa alkukohdejoukon solmuista taaksepäin aina ohjaavaan solmuun tai haarautumispisteeseen asti, kullekin solmulle lasketaan vastaavat tunnusluvut seuraavien sääntöjen mukaisesti:

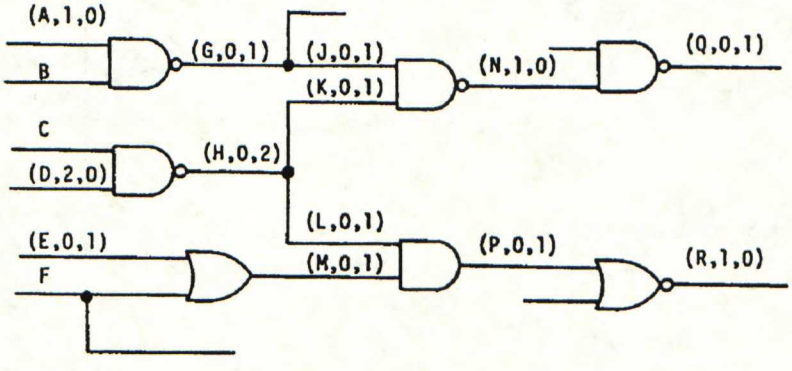
Sääntö 1 JA-portille: Olkoon X helpoimmin tilaan 0 ohjattava tulo ja Y portin lähtö. Silloin n0(X) = n0(Y) ja n1(X) = n1(Y). Muille tuloille Xi n0(Xi) = 0 ja n1(Xi) = n1(Y).

Sääntö 2 TAI-portille: Olkoon X helpoimmin tilaan 1 ohjattava tulo ja Y portin lähtö. Silloin n0(X) = n0(Y) ja n1(X) = n1(Y). Muille tuloille Xi n0(Xi) = n0(Y) ja n1(Xi) = 0.

Sääntö 3 invertterille: n0(X) = n1(Y) ja n1(X) = n0(Y).

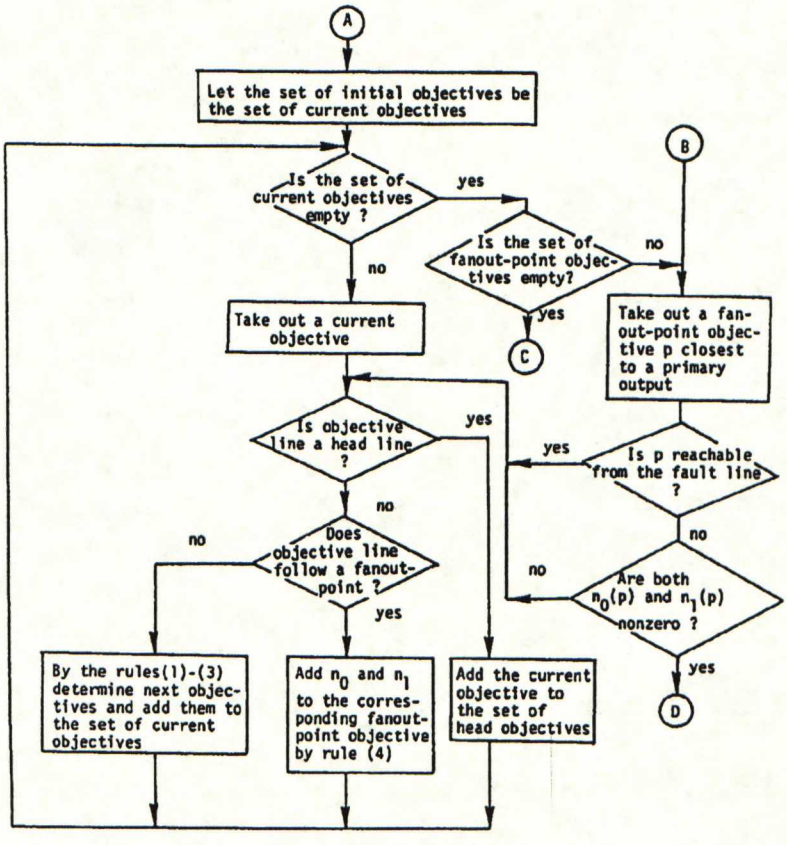
Sääntö 4 solmulle X, joka jakaantuu useaksi eri solmuksi X1, X2,...: n0(X) = n0(X1) + n0(X2) + n0(X3) + ... ja n1(X) = n1(X1) + n1(X2) + n1(X3) + ...

Näiden sääntöjen mukaisesti on saatu kuvassa 2.16 esitetyt solmujen tunnusluvut.



Kuva 2.16 Sääntöjen 1 - 4 mukaan lasketut solmujen tunnusluvut. [1, s.60]

Kuvassa 2.17 on esitettynä varsinainen monipolkuisen taaksepäin suuntautuvan etenemisen algoritmin vuokaavio. Aina kun saavutaan joko ohjaavaan solmuun tai haarautuvaan solmuun, kohdesolmu siirretään joko ohjaavien kohteiden tai haarautumiskohteiden joukkoon. Kun aktiivisten kohteiden joukko on tyhjä, otetaan haarautumiskohteiden joukosta haarautumissolmu, joka on lähinnä primäärituloa, ja määritellään sille arvo seuraavien ehtojen mukaan. Haarautumissolmu ei saa olla saavutettavissa vioitetusta solmusta normaaliin signaalin kulkusuuntaan edettäessä. Lisäksi n0 $\neq$ 0 ja n1 $\neq$ 0. Solmun tila olkoon 0, jos n0 > n1 ja 1, jos n1 > n0.



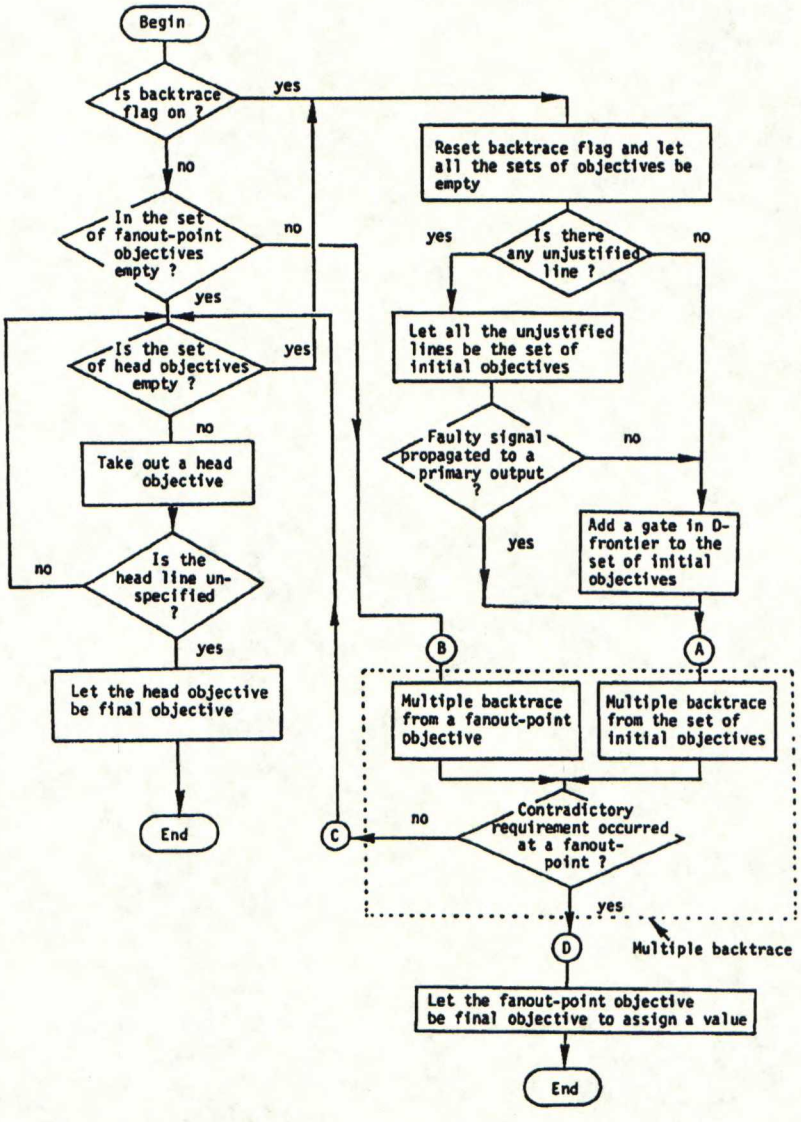
Kuva 2.17 Monipolkuinen taaksepäin suuntautuva algoritmi. [1, s.62]

Podemissa tilojen määrittelyjä sallitaan ainoastaan primäärituloissa. Fan-algoritmissa tilojen määrittelyjä sallitaan haarautumissolmuille sekä ohjaaville solmuille. Tällöin perääntymistä saattaa tapahtua haarautumissolmuissa ja ohjaavissa solmuissa, mutta ei primäärituloissa. Haarautumissolmulle määritellään tila siinä tapauksessa, että sille lasketut n0 ja n1 ovat molemmat nollasta poikkeavat eli on selvä mahdollisuus ristiriitaan. Koska haarautumissolmulle tehdään tilamääritys heti kun n0 ja n1 poikkeavat nollasta, paljastuu mahdollinen ristiriita välittömästi eikä turhaa työtä tehdä.

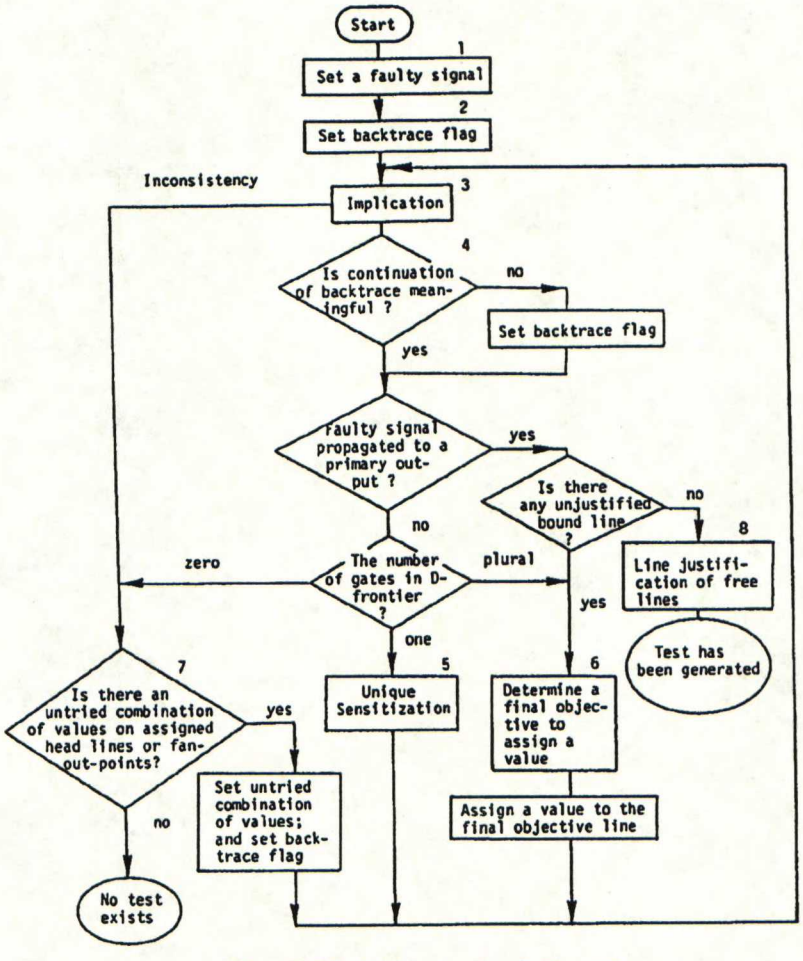
Ohje 6: Monipolkuisen taaksepäinsuuntautuvan etenemisen aikana seurataan kohteena olevan haarautumissolmun tunnuslukuja n0 ja n1 siten, että jos molempien arvo poikkeaa nollasta, keskeytetään eteneminen siihen ja määritetään solmulle tila.

Mikäli jompikumpi tunnusluvuista pysyy nollana, jatketaan etenemistä taaksepäin, kunnes saavutetaan ohjaava solmu. Jos kaikki kohteet päätyvät lopuksi ohjaaviin solmuihin eli sekä haarautumissolmujen että aktiivisten solmujen joukot ovat tyhjiä, siirrytään antamaan tiloja ohjaaville solmuille yksi kerrallaan tehden aina tarkistussimuloinnin.

Kuvassa 2.19 on esitetty koko Fan-algoritmin vuokaavio, johon liittyvä lopullisen kohteen valintamenetelmä on esitetty kuvassa 2.18. Varsinainen monipolkuinen taaksepäinsuuntautuva etenemisalgoritmi on esitetty kuvassa 2.17.



Kuva 2.18 Lopullisen kohteen valitseminen. [1, s.65]



Kuva 2.19 FAN-algoritmi. [1, s.63]

Pääalgoritmin eri vaiheiden toiminta on seuraava (numero viittaa vastaavaan vaiheeseen vuokaaviossa) :

1. Määritellään vioitettava solmu ja vian tyyppi eli solmuun joko tila D tai /D.

2. Varsinainen etenemisalgoritmi voi alkaa kahdesta eri kohdasta. Tässä asetettavalla lipulla määritellään aloitetaanko toiminta kohdasta A vai B sen mukaan, onko kysymyksessä alkukohde vai haarautumiskohde. Asettamaton lippu ohjaa alkukohteeseen eli kohtaan A.

3. Tässä vaiheessa määritellään niin monta signaalia yksikäsitteisesti tiettyyn tilaan kuin vain mahdollista. Kysymyksessä on toiminto, joka suorittaa eräänlaisen logiikkasimuloinnin sekä eteenpäin että taaksepäin. Koska tilojen määrittelyssä kuljetaan myös signaalien normaalia kulkusuuntaa vasten, on mahdollista, että jotkut signaalit jäävät määrittelemättä. Monipolkuista etenemisalgoritmia voidaan käyttää myös tässä vaiheessa määrittelemään puuttuvat signaalitilat.

4. Tarkistetaan monipolkuetenemisen tarpeellisuus. Etenemistä ei katsota tarpeelliseksi, jos edellisen kohteen tarkoituksena oli siirtää tila D tai /D eteenpäin ja D-rintama on muuttunut. Myöskään eteneminen ei ole tarpeellista, jos edellisen kohteen tarkoitus oli määritellä määrittelemättömille signaaleille tilat, ja nyt määrittelemättömiä tiloja ei enää ole. Jos todetaan, ettei eteneminen ole tarpeellista, poistetaan monipolkuetenemisen lippu eli ohjataan seuraava eteneminen alkamaan alkukohteesta.

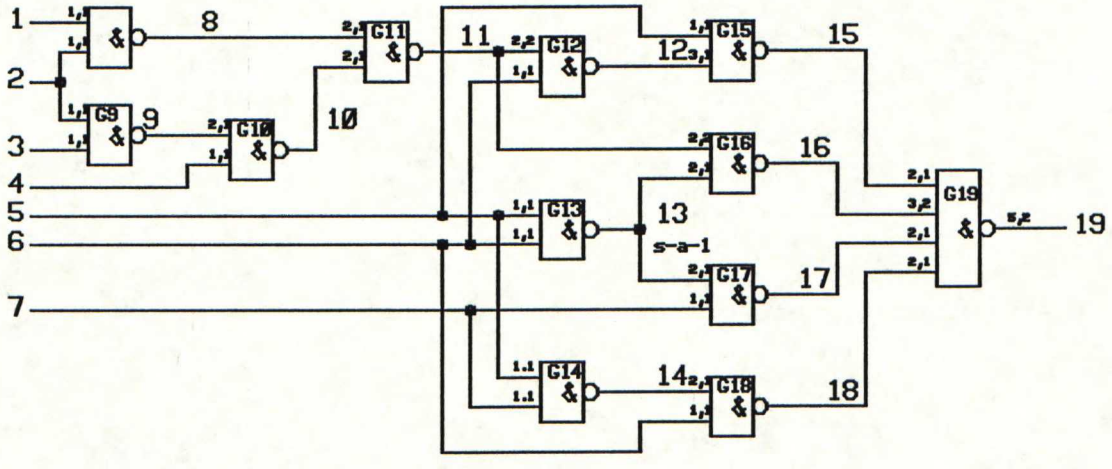
5. Suoritetaan herkistys ohjeen 3 mukaisesti.

6. Loppukohteen määrittely. Vaiheen algoritmi on kuvassa 2.18. Tarkoituksena on valita solmu ja sille tila siten, että autetaan mahdollisimman paljon alkukohteiden vaatimusten toteuttamisessa.

7. Perääntyminen on toteutettu samoin kuin Podem-algoritmissa. Muodostetaan tehdyistä tilamäärittelyistä kronologinen lista, josta ilmenee solmun tunniste ja annettu tila. Jos perääntymisen yhteydessä tila vaihdetaan, siitä ilmoittamaan asetetaan merkkilippu.

8. Vapaiden solmujen vaatimusten täyttäminen. Kuten aiemmin ohjeessa 4 todettiin, Fan-algoritmissa keskeytetään taaksepäinsuuntautuva eteneminen heti kun kohdataan ohjaava solmu. Tämä vaihe suorittaa vaatimusten täyttämisen loppuun. Peräytymiseen ei tässä vaiheessa jouduta ohjeessa 4 mainituista syistä.

2.6.3 Esimerkki



Kuva 2.20 Kytkentä Fan-algoritmin havainnollistamiseksi.

Esimerkkinä käytetään samaa kytkentää kuin D-algoritmin ja Podem-algoritmin esimerkeissä. Tässä tapauksessa kytkentään on lisäksi merkitty jokaisen tulon viereen lukupari, joka kuvaa sen ohjattavuutta nollaan ja ykköseen siten, että ensimmäinen luku kuvaa ohjattavuutta nollaan ja toinen ohjattavuutta ykköseen. Mitä pienempi luku on, sitä parempi on ohjattavuus.

Alkuun määritellään viaksi solmuun 13 s-a-1 eli oikosulku käyttöjännitteeseen.

Seuraavaksi suoritetaan eräänlainen simulointi vikakohdasta sekä eteenpäin että taaksepäin siten, että määritellään niin monta signaalia kuin vain yksikäsitteisesti nyt vallitsevan tilanteen mukaan voidaan määritellä.

Tulokseksi saadaan: 13 = /D, 5 = 1, 6 = 1. (Merkintätapa: 'solmu = tila')

Todetaan, että D-rintamalla on kaksi porttia, joten siirrytään loppukohteen määrittelyyn.

Lisätään yksi D-rintaman porteista alkukohteiden joukkoon eli valitaan G16, jonka lähtöön pitäisi saada D.

Ryhdytään monipolkuetenemiseen, jonka tuloksia ovat: ($G16_{input1}$,0,1), (11,0,1), (8,1,0), (10,0,0), (1,0,1), ($G8_{input2}$,0,1), (2,0,1). Lisäksi ohjaavien kohteiden joukkoon jää (2,0,1).

Ohjaavasta kohteesta (2,0,1) tulee loppukohde ja sille määritellään tila 1.

Palataan takaisin alkuun ja suoritetaan yksikäsitteisten tilojen määrittely jälleen, jonka tuloksena saadaan: 13 = /D, 5 = 1, 6 = 1 ja 1 = 1.

Todetaan, että D-rintamalla on kaksi porttia.

Otetaan loppukohteeksi ohjaavien kohteiden joukosta yksi kohde eli (2,0,1) ja määritellään sille tila eli 2 = 1.

Palataan alkuun ja suoritetaan yksikäsitteisten tilojen määrittely, jolloin saadaan: 1 = 1, 2 = 1, 5 = 1, 8 = 0, 11 = 1, 12 = 0, 13 = /D, 15 = 1 ja 16 = D.

Todetaan, että D-rintamalla on kaksi porttia.

Otetaan D-rintamalta yksi portti alkukohteeksi eli G17:n lähtöön halutaan D.

Ryhdytään monipolkuetenemiseen, jonka tuloksia ovat: ($G17_{input2}$,0,1), (7,0,1), jotka siirretään haarautumiskohteiden joukkoon. Sama kohde otetaan pois haarautumiskohteiden joukosta, todetaan sen olevan ohjaava solmu ja siirretään se ohjaavien kohteiden joukkoon.

Määritellään kohde ohjaavien kohteiden joukosta eli annetaan solmulle 7 tila 1.

Suoritetaan yksikäsitteisten tilojen määrittely, jolloin tuloksena saadaan:
1 = 1, 2 = 1, 5 = 1, 6 = 1, 7 = 1, 8 = 0, 11 = 1, 12 = 0, 13 = /D, 14 = 0, 15 = 1, 16 = D, 17 = D, 18 = 1 ja 19 = /D.

Todetaan, että /D on edennyt primäärilähtöön ja että kaikkien sidottujen solmujen tilat on määritelty. Määritellään lopuille vapaille solmuille eli solmuille 3 ja 4 tilat esimerkiksi tila 1 kummallekin ja todetaan testi tehdyksi.

2.6.4 Ominaisuuksia

Kuten esimerkkitapauksestakin voidaan todeta, Fan-algoritmi on älykkäämpi kuin esimerkiksi Podem. Se generoi testin esimerkkikytkennälle ilman peräänty-mistä, kun taas D-algoritmi ja Podem joutuivat kumpikin perääntymään ker-ran. Fan-algoritmi vaatii kuitenkin huomattavasti enemmän esityötä, koska kytkennästä on määriteltävä vapaat solmut, ohjaavat solmut, sidotut solmut se-kä haarautumissolmut. Lisäksi on ennen varsinaisen generoinnin aloitusta las-kettava kullekin solmulle ohjattavuuden tunnusluvut sekä tilaa 1 että 0 varten. Algoritmeja vertaillessa on kuitenkin todettu, että Fan-algoritmi on muutaman kerran Podem-algoritmia nopeampi. Nopeusero riippuu selvästi kytkennän tyy-pistä.

2.7 KRIITTINEN POLKU

2.7.1 Yleistä

Kriittisen polun algoritmi eli 'Critical Path - algorithm' on kehitetty D-algoritmista. Siinä ei pyritä sadan prosentin kattavuuteen, vaan mahdollisimman vähällä työllä mahdollisimman suureen kattavuuteen. Menetelmää voidaan käyttää esimerkiksi karkeana testingeneroinnin aloitustyökaluna tai piirin testattavuuden tarkastamiseen.

2.7.2 Toimintaperiaate [11]

Algoritmi aloittaa määrittelemällä aloituskohteeksi yhden primäärilähdön ja sille tilan.

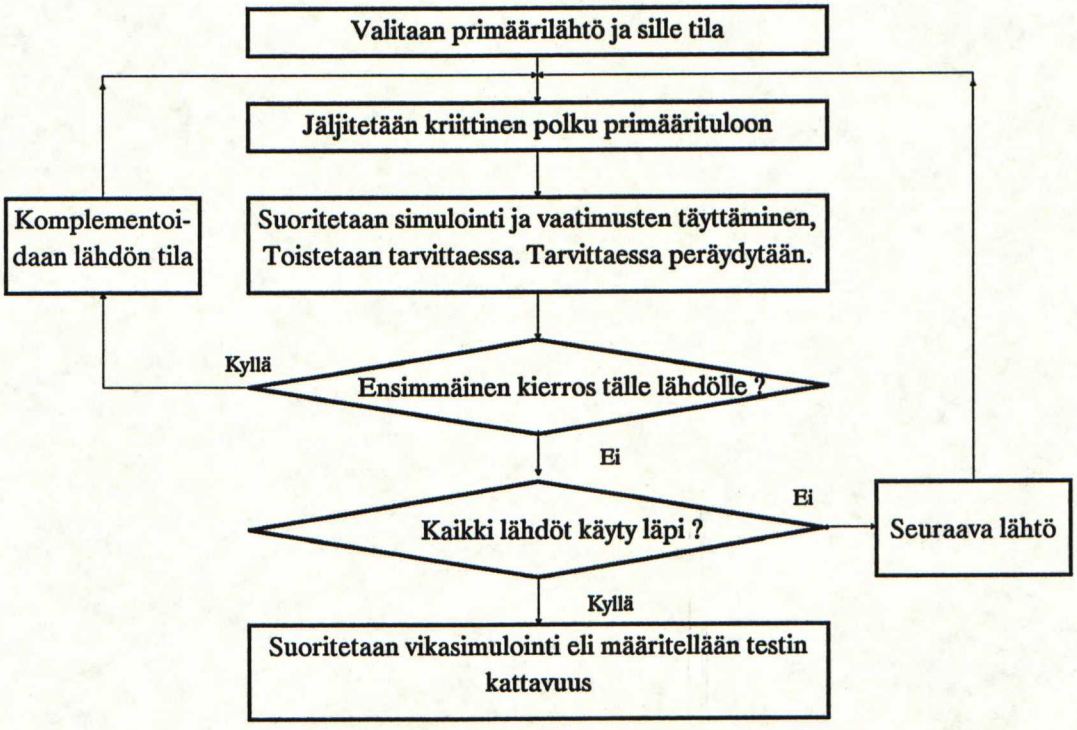
Seuraavaksi ryhdytään kriittisen polun määrittelyyn, jossa kuljetaan koko piirin läpi kriittistä polkua pitkin primäärilähdöstä aina primärituloon asti. Solmun katsotaan kuuluvan kriittiselle polulle, jos se on sellaisen portin tulo, jonka lähtö on kriittisellä polulla, ja jos sen tilan komplementointi aiheuttaa mainitun portin lähdön tilan komplementoitumisen. Piirin primäärilähtö on aina kriittisellä polulla. Algoritmi seuraa kriittistä polkua määritellen samalla kunkin portin muille tuloille tarvittavat tilat. Jos kriittinen polku haarautuu useampaan haaraan, tehdään haarojen välillä satunnainen valinta.

Kun primäritulo saavutetaan, suoritetaan vaatimusten täyttäminen esimerkiksi D-algoritmista tuttuja tauluja hyväksi käyttäen. Jos syntyy ristiriita, peräydytään edellisistä algoritmeista tuttuun tapaan vaiheeseen, jossa viimeksi tehtiin satunnainen valinta, vaihdetaan valinta toiseksi ja yritetään uudelleen.

Lopuksi vaihdetaan valitun primäärilähdön tila komplementiksi ja suoritetaan vastaavat vaiheet uudelleen.

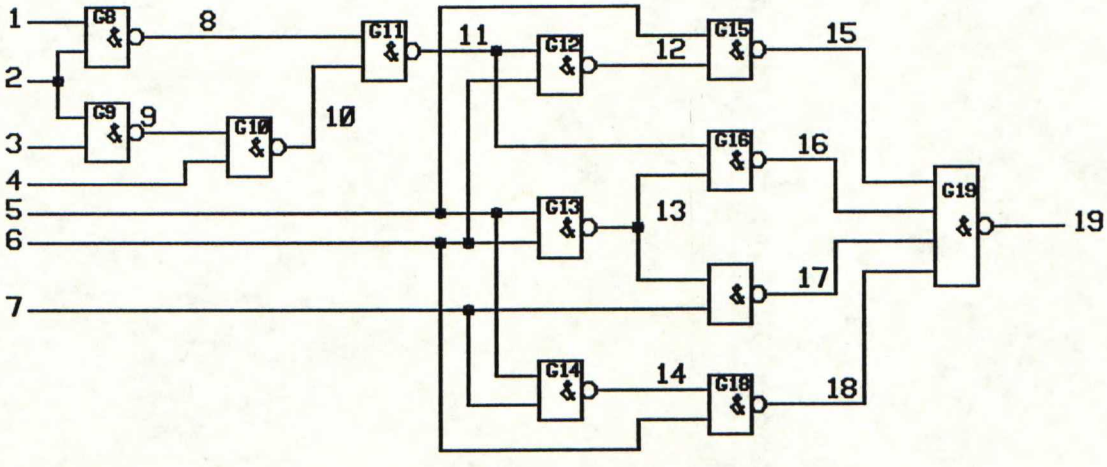
Seuraavaksi suoritetaan vikasimulointi ja saadaan kattavuus.

Näin toimien käydään kaikki piirin primäärilähdöt läpi ja saadaan testiohjelma tuotetuksi.



Kuva 2.21 Kriittisen polun algoritmin vuokaavio.

2.7.3 Esimerkki



Kuva 2.22 Kytkentä kriittisen polun algoritmin havainnollistamiseksi.

Valitaan käsiteltäväksi lähdöksi solmu 19 ja sille tila 0.

Lähdön tila on 0, jos kaikkien portin 19 tulojen tila on 1. Valitaan kriittiselle polulle satunnaisesti solmu 15 ja sille siis tila 1. Solmuille 16, 17 ja 18 asetetaan vaatimukseksi tila 1.

Määritellään solmu 12 kriittiselle polulle ja sille tila 0, jolloin solmun 5 tilan pitää olla 1.

Määritellään solmu 11 kriittiselle polulle ja sille tila 1, jolloin solmun 10 tilan pitää olla 1.

Määritellään solmu 8 kriittiselle polulle ja sille tila 0, jolloin solmun 2 tilan pitää olla 1.

Todetaan, että on saavutettu primääritulo eli solmu 1, jonka tila on 1.

Suoritetaan simulointi ja todetaan, ettei ristiriitoja syntynyt. Kuitenkaan kaikkia vaatimuksia ei ole täytetty.

Täytetään vaatimus, joka määrittää solmulle tilan 1. Koska solmun 11 tila on jo 1, asetetaan solmun 13 tilaksi 0, joka puolestaan edellyttää solmulle 5 tilaa 1.

Koska on jälleen saavutettu primääritulo, suoritetaan simulointi.

Edelleen on täyttämättä solmun 18 tilan 1 vaatimukset. Koska solmu 6 on jo tilassa 1, on solmu 14 saatava tilaan 0. Se edellyttää solmulle 7 tilaa 1.

Suoritetaan simulointi ja todetaan lähdön asettuneen haluttuun tilaan.

Vaihdetaan lähdön tilaksi 1.

Määritellään kriittiselle polulle solmu 18 ja sille tilaksi 0. Vaatimuksiksi saadaan solmuille 15, 16 ja 17 tila 1.

Määritellään solmu 14 kriittiselle polulle ja sille tila 1, jolloin solmun 6 tilan pitää olla 1.

Määritellään solmu 5 kriittiselle polulle ja sille tila 0, jolloin solmun 7 tilan pitää olla 1.

Koska on saavutettu primääritulo, suoritetaan simulointi, jonka tuloksena todetaan, että on syntynyt ristiriita simuloinnin ja aiemmin asetettujen tilojen välillä. Simulointi antoi solmulle 17 tilan 0, kun taas vaatimuksissa sen tilaksi määriteltiin tila 1. Nyt joudutaan peräytymään eli vaihdetaan edellinen satunnaisesti tehty valinta toiseksi. Sen sijaan, että määriteltiin solmu 5 kriittiselle polulle, määritellään solmu 7 kriittiselle polulle ja sille tila 0. Vaatimukseksi saadaan solmulle 5 tila 1.

Suoritetaan simulointi ja todetaan, että kaikkia vaatimuksia ei ole vielä täytetty. Koska solmun 15 vaatimuksena on tila 1 ja solmu 5 on tilassa 1, määritellään solmu 12 tilaan 0.

Määritellään solmu 11 tilaan 1, solmu 8 tilaan 0 ja solmut 1 ja 2 tilaan 1.

Suoritetaan simulointi ja todetaan, että lähtö on asettunut haluttuun tilaan.

Koska kaikki lähdöt on nyt käyty läpi, suoritetaan vikasimulointi muodostetuilla vektoreilla ja saadaan kattavuudeksi 50%.

2.7.4 Ominaisuuksia

Kriittisen polun algoritmi on approksimoiva, eikä se pysty tuottamaan testiä kaikille vioille. Ongelmia aiheuttavat erityisesti haarautuvat ja uudelleen yhtyvät signaalitiet. Koska algoritmi tekee runsaasti satunnaisia valintoja, on erittäin todennäköistä, että jossain vaiheessa joudutaan peräytymään. Kuitenkin algoritmin yksikertaisuuden vuoksi laskennan määrä on merkittävästi pienempi kuin esimerkiksi D-algoritmissa.

2.8 MARLETT-ALGORITMI

2.8.1 Yleistä

Marlett-algoritmin kehitti R. Marlett 70- ja 80-luvun vaihteessa. Se on yksittäisen polun herkistämiseen perustuva algoritmi, joka etenee kytkennässä taaksepäin sekä ajassa että solmutasolla. Algoritmi on kehitetty erityisesti sekvenssilogiikan testivektoreiden generointiin, ja se pystyy tuottamaan testin vaikka takaisinkytkentöjä sisältävän kytkennän sekvenssipiirit eivät olisi edes nollattavissa

2.8.2 Toimintaperiaate [4, s.69-70]

Algoritmi aloittaa valitsemalla vian, jolle ei ole vielä tehty testiä. Sen jälkeen Scoap-tyyppisen testattavuusanalyysin tuottaman tiedon avulla valitaan primäärilähtö, jonka kautta vioitettavan portin vikavaikutus näkyy helpoimmin ulospäin.

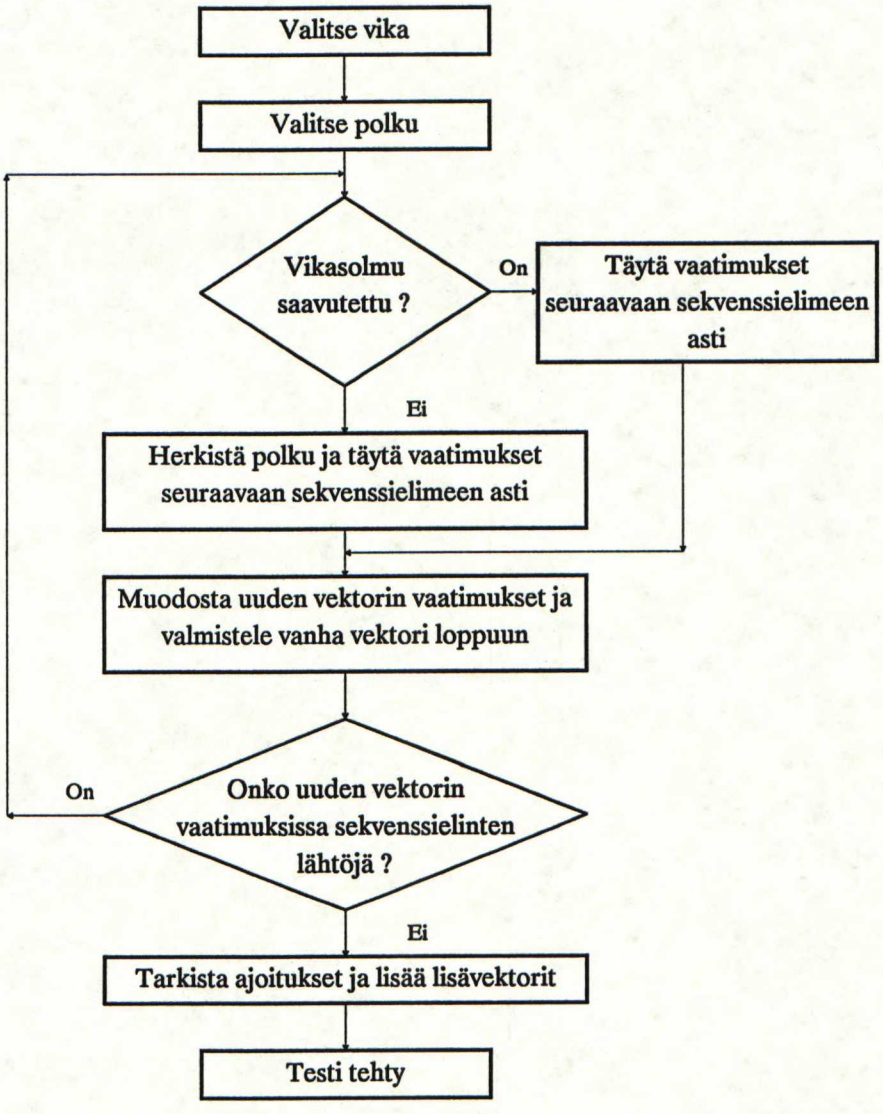
Valitusta lähdöstä jatketaan kohti vioitettavaa solmua herkistämällä polkua, kunnes saavutetaan vioitettava solmu. Vioitettavan solmun jälkeen tavoite muuttuu vian herättämiseksi.

Testiä generoidaan vektori vektorilta siten, että aloitetaan viimeisestä vektorista ja lopetetaan ensimmäiseen vektoriin. Algoritmi etenee siis sekä kytkennässä että ajassa taaksepäin.

Kun vaatimuksia täytettäessä kohdataan tilanne, jossa sekvenssipiirin lähtöjen on oltava tietyssä tilassa, täytetään vaatimukset kombinaatiologiikan osalta ja tehdään uusi vektori, jonka vaatimuksina on edellämainittujen lähtöjen tilat. Vaatimuksia uudelle vektorille synnyttävät heti myös sekvenssipiirien mahdolliset kellosignaalit, joissa tarvitaan esimerkiksi nouseva reuna, jotta tulon tila siirtyisi tulosta lähtöön.

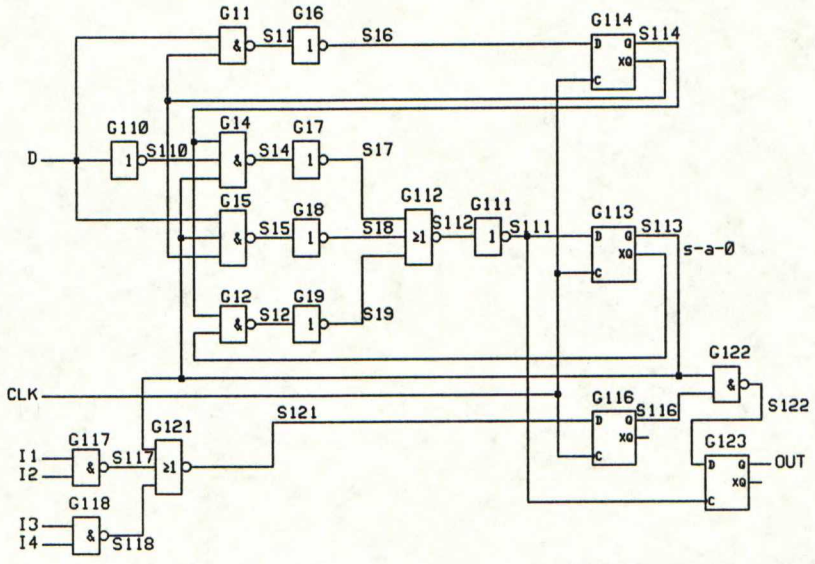
Mahdolliset ristiriitatilanteet aiheuttavat perääntymisen, jossa palautetaan tilanteeksi viimeisin valintatilanne, ja tehdään uusi valinta. Valinnat esimerkiksi signaalitien suhteen perustuvat Scoap-tyyppiseen testattavuusanalyysiin. Valinnoissa noudatetaan samaa logiikkaa kuin Podem-algoritmissa esimerkiksi valittaessa se portin tulo, jonka tilan vaatimusten täyttämistä yritetään ensimmäiseksi.

Polun herkistystä tai vian herättämistä niihin liittyvine vaatimusten täyttöineen jatketaan. Tarvittaessa luodaan uusia vektoreita, kunnes saavutetaan tilanne, jossa kaikki vaatimukset on täytetty, mutta sekvessielinten lähdöille ei enää ole vaatimuksia.



Kuva 2.23 Marlett-algoritmin vuokaavio.

2.8.3 Esimerkki



Kuva 2.24 Kytkentä Marlett-algoritmin havainnollistamiseksi.

Ohj. '0':aan	'1':een
CLK 1	1
D 1	1
I1 1	1
I2 1	1
I3 1	1
I4 1	1
OUT 35	79
S11 15	1
S12 48	14
S14 51	1
S15 51	1
S16 10	10
S17 1	51
S18 1	51
S19 14	48
S110 1	1
S111 36	36
S112 48	16
S113 35	40
S114 14	14
S116 5	43
S117 2	1
S118 2	1
S121 1	39
S122 83	5

Taulukko 2.3 Scoap-tyyppisen testattavuusanalyysin ohjattavuustulokset.

Tuotetaan kuvan 2.24 kytkennän solmun S113 vialle s-a-0 testi. Testattavuustietojen pohjalta todetaan parhaaksi poluksi vikautettavasta solmusta primäärilähtöön olevan S113 - S122 - OUT.

Ensin siis pyritään herkistämään polku vikautettavasta solmusta primäärilähtöön.

Jotta signaali kulkisi OUT-lähtöön, on kiikun G123 kellotuloon tultava nouseva reuna eli tällä vektorille tila 1 eli S111 = 1. (Merkintä: signaalinimi = tila)

Siitä seuraa: S112 = 0, taulukon 2.3 mukaan S19 on helpoimmin ohjattavissa 1:een eli S12 = 0, jolloin S114 = 1 ja S113 = 0. Kaksi viimeistä vaatimusta ovat uuden vektorin tulosvaatimuksia, koska ne ovat sekvenssipiirin lähtöjä. Lisäksi nousevan reunan tuottamiseksi signaaliin S111 on S111:n oltava tilassa 0 edellisessä vektorissa, eli edellisen vektorin lähtövaatimuksiin kuuluu S111 = 0. Signaalissa CLK on myös täytynyt olla nouseva reuna, joten tälle vektorille on vaatimus CLK = 1 ja uudelle vektorille CLK = 0.

Lähtövaatimuksina ovat siis uudelle vektorille S16 = 1, S111 = 0 sekä CLK = 0.

Seuraukset: Koska S16 = 1, on S11 = 0 eli D = 1 ja S114 = 0. Koska S111 = 0, on S112 = 1, S17 = S18 = S19 = 0 ja S14 = S15 = S12 = 1. Koska D = 1 eli S110 = 0, on S14 vaatimukset täytetty. Koska S114:lle asetettu uusi vaatimus on tila 0, on myös S12:n vaatimukset täytetty. S15:n vaatimusten täyttöön jää ainoastaan yksi mahdollisuus eli S113 = 0. Lisäksi tällä vektorilla on S116:n tilan oltava 1, jotta määritelty polku olisi herkistetty S111:n nousevalla reunalla.

Koska G113:n, G116:n ja G114:n lähtöihin on saatava tietyt tilat, muodostuu CLK:n vaatimuksiksi tälle vektorille tila 0 ja uudelle vektorille tila 1.

Uuden vektorin vaatimus siis on yksinkertaisesti CLK = 1, koska sekvenssipiirin tuloille ei kannata esittää vaatimuksia, jos kellosignaalin seuraava reuna ei ole aktiivinen.

Seuraavalle vektorille syntyvät vaatimukset siis ovat : CLK = 0, S16 = 0, S111 = 1 ja S121 = 1.

Tästä seuraa: S11 = 1, D = 0. S112 = 0, S19 = 1, S12 = 0, S113 = 0 ja S114 = 1. S121:n osalta seuraa: S113 = 0, S117 = 0, S118 = 0, I1 = I2 = I3 = I4 = 1.

Koska G113:n ja G114:n lähtöihin on saatava tietyt tilat, muodostuu CLK:n vaatimukskesi uudelle vektorille tila 1.

Uuden vektorin vaatimus on yksinkertaisesti CLK = 1 kuten edelläkin.

Seuraavalle vektorille syntyneet vaatimukset ovat: CLK = 0, S16 = 1 ja S111 = 0.

S16:n osalta seuraa: S11 = 0, D = 1 ja S114 = 0. S111:n osalta seuraa: S112 = 1, S17 = S18 = S19 = 0, S14 = S15 = S12 = 1. Koska D = 0 ja S114 = 0, ovat vaatimukset täytetyt.

Koska G113:n ja G114:n lähtöihin on saatava tietyt tilat, muodostuu CLK:n vaatimuksiksi uudelle vektorille tila 1.

Uuden vektorin vaatimus siis on yksinkertaisesti CLK = 1 kuten edelläkin.

Seuraavan vektorin vaatimuksiksi muodostuvat: CLK = 0, S16 = 0. Siitä seuraa: S11 = 1, D = 0.

Koska sekventiaalisten elinten lähdöille ei enää ole mitään määrittelyjä, kaikki vaatimukset on täytetty eikä ristiriitoja ole, on testi tuotettu.

Loppukäsittelyssä tarkistetaan, ettei minkään sekventiaalisen elimen tulojen tilat muutu kellon aktiivisella reunalla, lisätään tarvittaessa täytevektoreita ja määritellään tilat sellaisille tuloille, joilla ei ole merkitystä testin kannalta.

Valmis testi on seuraava:

D	CLK	I1	I2	I3	I4
0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	1	0	0	0	0

2.8.4 Ominaisuuksia

Marlett-algoritmin oleellinen etu on sen tapa käsittellä sekvenssilogiikkaa. Koska algoritmi ei pyri ennustamaan tulevaisuutta, vaan asettaa vaatimuksia menneelle ja etenee ajassa taaksepäin, se pystyy tuottamaan testin kaikille sekvenssilogiikkaa ja kombinaatiologiikkaa sisältäville kytkennöille, mikäli testi on yleensä toteutettavissa. Marlett-algoritmia on käytetty kahden kaupallisen generointiohjelmiston pohjana. Algoritmin yksittäisen polun herkistäminen laajenee usean polun samanaikaiseksi herkistykseksi haarautuvien solmujen ja uudelleen yhtyvien signaaliteiden yhteydessä. Lisäksi algoritmi sisältää eräänlaisen ristiriita-analysaattorin perääntymistarpeen vähentämiseksi.

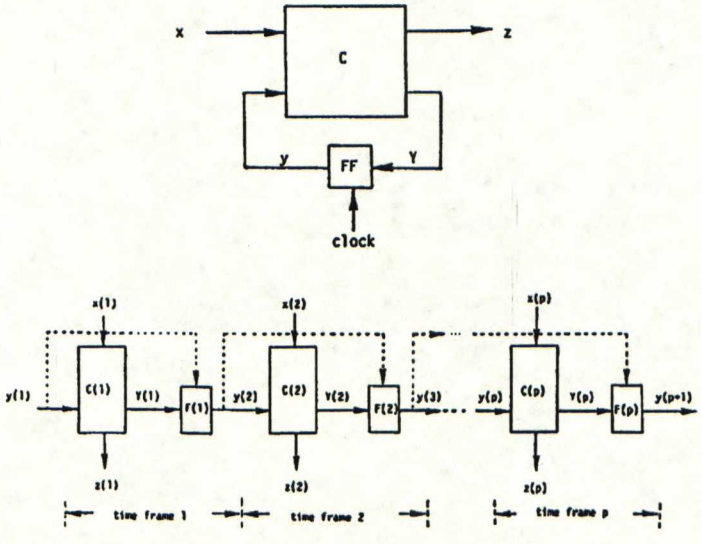
2.9 LAAJENNETTU D-ALGORITMI

2.9.1 Yleistä

Laajennetun D-algoritmin kehitti vuonna 1968 Kubo. Se pystyy tuottamaan testin synkroniselle sekvenssilogiikalle, joten se on huomattavasti monikäyttöisempi kuin tavallinen D-algoritmi. Lähes samanlainen menetelmä on myös yhdeksäntilainen algoritmi, jonka kehitti Moth vuonna 1976.

2.9.2 Toimintaperiaate [1, s.67-74]

Laajennetussa D-algoritmissa kytkennästä muodostetaan kytkentäketju, jossa kombinaatiologiikka toistuu peräkkäin kytkettynä niin monta kertaa, kuin testissä tarvitaan aktiivisia kelloreunoja. Kuva 2.25 havainnollistaa asiaa. Kuvan yläosassa on alkuperäinen kytkentä kombinaatio-osa ja sekvenssiosa eroteltuna, ja alaosassa on kolmekertaisesti monistettu kytkentä, jossa sekvenssipiirit on poistettu siten, että kunkin sekvenssipiirin tulosolmu on kytketty seuraavan kopion sekvenssipiirin lähtösolmuun. Varsinainen testingenerointi palautuu siis kombinaatiologiikan testingeneroinniksi.

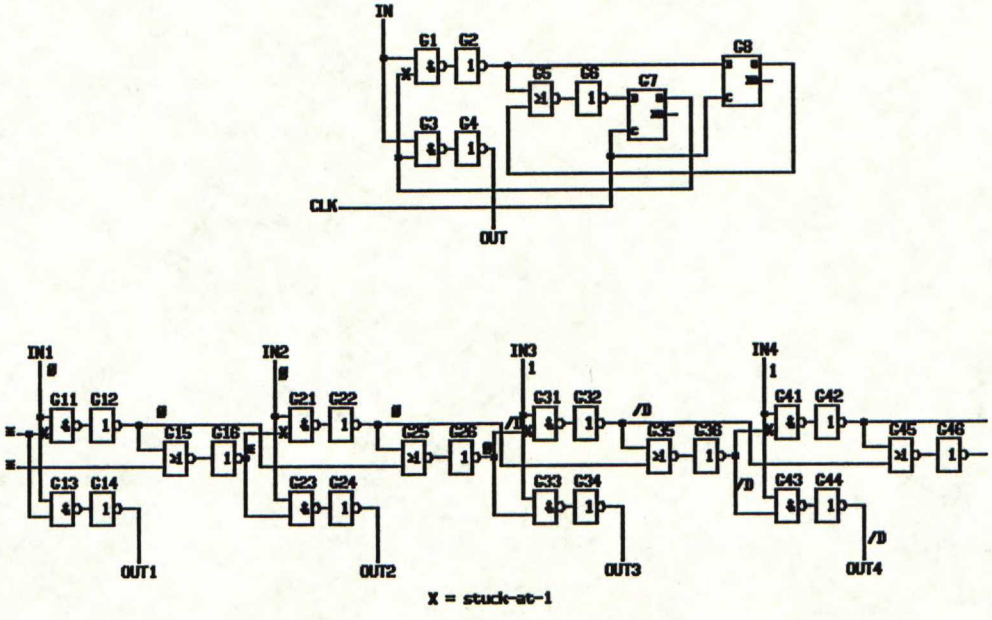


Kuva 2.25 Sekvenssipiirin muuttaminen kombinaatiokytkentöjen ketjuksi. [1, s.72]

Jokainen monistettu kombinaatiokytkentä vastaa yhtä kelloreunaa, eli ne sijaitsevat tavallaan eri ajassa siten, että ensimmäisen kopion tulosignaalien aiheuttamat lähtösignaalit eli sekvenssipiirien tulosignaalit siirtyvät ensimmäisellä kelloreunalla seuraavan kopion tulosignaaleiksi.

Varsinaisena testingenerointialgoritmina toimii tavallinen D-algoritmi.

2.9.3 Esimerkki



Kuva 2.26 Kytkennät laajennetun D-algoritmin havainnollistamiseksi.

Tuotetaan testi kuvan 3.26 yläosan esittämälle kytkennälle, jossa vikana on portin G1 alemmassa tulossa s-a-1.

Monistetaan kytkentää neljä kertaa. Käytännön algoritmissa monistusta voidaan suorittaa aina tarvittaessa. Arvioidaan, että vika saadaan herätetyksi kolmannessa kopiossa, eli portin G31 alemmalle tulolle asetetaan tavoite /D. On muistettava, että vastaavalle solmulle aiemmissa kopioissa ei saa asettaa vaatimukseksi kuin tila 1 tai määrittelemätön tila. Monistuksen jälkeen kytkentä on kuvan 3.26 alaosan mukainen.

Kopioidun kytkennän portin G31 alemmalle tulolle vaaditaan siis tila 0, jotta vika saataisiin herätetyksi. Tämä tarkoittaa tilaa 0 solmussa G26, joka vastaavasti edellyttää tilaa 0 solmuissa G22 ja G12. G12 on puolestaan tilassa 0, jos tulo IN1 on tilassa 0. Vastaavasti G22 on tilassa 0, jos tulo IN2 on tilassa 0.

Vian eteneminen primäärilähtöön edellyttää tuloon IN3 tilaa 1 ja tuloon IN4 myös tilaa 1. Muille signaaleille ei aseteta vaatimuksia. Koska ensimmäisen kopion näennäisille kahdelle tulolle, jotka on merkitty *:llä, ei syntynyt vaatimuksia ja /D on edennyt primäärilähtöön OUT4, voidaan todeta testi tehdyksi, jolloin testisekvenssi on:

IN	CLK
0	0
0	1
0	0
0	1
1	0
1	1
1	0
1	1

2.9.4 Ominaisuuksia

Laajennetulla D-algoritmilla ei voida tuottaa kovin pitkiä vektorijoukkoja, koska jokainen testijakso synnyttää uuden kopion kytkennästä ja siten saattaa vaatia hyvinkin runsaasti muistia. Laajennettu D-algoritmi edellyttää myöskin toimiakseen täydellisesti täysin synkronista logiikkaa. Laajennetun D-algoritmin pohjalta on kuitenkin kehitetty algoritmi, joka tuottaa testiehdotuksen asynkroniselle kytkennälle. Todellinen toimivuus tarkistetaan kuitenkin erikseen vikasimulaattorilla. Laajennettu D-algoritmi soveltuu erityisen hyvin dynaamiselle logiikalle, koska toteutusmenetelmästä johtuen kellosignaalit askeltavat koko ajan.

3. KAUPALLISET ATPG-ALGORITMIT, NIIDEN SOVELTUVUUS JA LIITETTÄVYYS

3.1 NEXTGEN

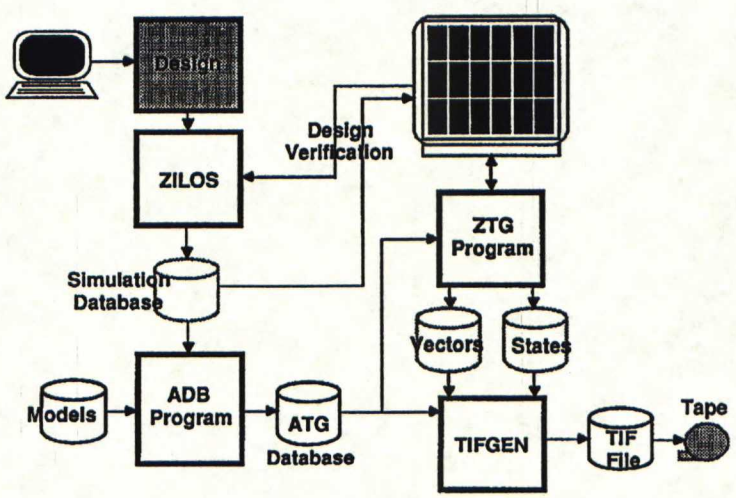
3.1.1 Yleistä [8]

Nextgen on Zycad Corp:n kehittämä testivektorigeneraattori, joka perustuu sekä Marlett-algoritmiin että Zycad Corp:n vikasimulointikiihdyttimeen. Se tuottaa automaattisesti testin sekä kombinaatiologiikalle että sekvenssilogiikalle tai niiden yhdistelmälle. Nextgen käyttää samaa kytkentälistaa ja mallitustapaa kuin logiikkasimulaattori Zilos, joka puolestaan on lähes täysin yhteensopiva Micronas Oy:n käyttämän Silos-simulaattorin kanssa.

Nextgen sallii käyttäjälle mahdollisuuden vaikuttaa testingenerointiin asettamalla suurimman sallitun CPU-ajan ja suurimman sallitun perääntymismäärän vikaa kohden. Lisäksi Nextgeniä voidaan käyttää interaktiivisesti, jolloin käyttäjä voi määritellä mm., missä järjestyksessä vioille tehdään testit. Generoinnin voi keskeyttää milloin vain, ja sitä voi jatkaa samasta vaiheesta myöhemmin. Vastaavasti tahattomasti keskeytynyt ajo voidaan aloittaa keskeytymisvaiheesta uudelleen.

Järjestelmään kuuluu testattavuusanalyysi, jonka avulla voidaan tarkastella suunnittelun soveltuvuutta automaattiseen testingenerointiin, ennen kuin varsinainen generointi aloitetaan.

Generoinnin tuloksena saadaan testiherätteet ja niitä vastaavat vasteet, vikakirjasto ja piirin sisäiset tilat vianetsintää varten, muistien alustusdata, signaalinimien ristiinviittaustaulukko ja havaitsemattomien vikojen lista sekä testattavuusanalyysin tulokset.



Kuva 3.1 NextGen kokonaisjärjestelmä. [5, s. 3-1]

3.1.2 Soveltuvuus ja liitettävyys

Micronas Oy:ssä käytetään logiikkasimulointiin pääasiassa Silos-logiikka- ja vikasimulaattoria. Koska Silos ja Zilos ovat lähes täysin yhteensopivia, olisi Nextgenin liitettävyys nykyiseen suunnittelujärjestelmään kohtuullisen hyvä. Koska Nextgen on tarkoitettu sekä kombinaatiologiikan että sekvenssilogiikan testingenerointiin, olisi se myös siinä suhteessa soveltuva Micronasin suunnittelujärjestelmän osaksi. Ohjelmistosta on saatavilla Micronasin VAX-järjestelmään sopiva versio sekä versiot Daisy-, Mentor- ja Tegas-työasemiin. Primitiiveihin kuuluvat perusportit, siirtoportit, NMOS- ja PMOS-kytkimet sekä RAM ja ROM.

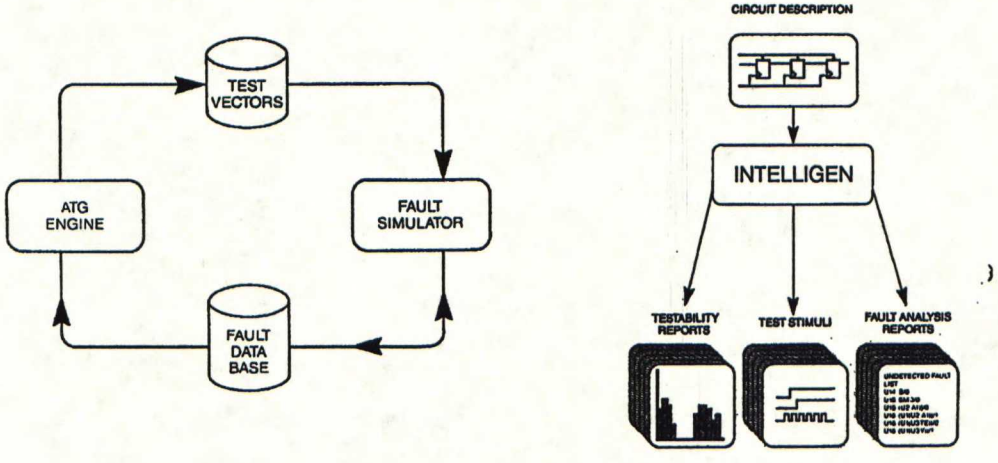
Haittatekijänä mainittakoon koko järjestelmän korkea hinta 410 000 USD (-88).

3.2 INTELLIGEN

3.2.1 Yleistä [7]

Intelligen on HHB Systemsin kehittämä testivektorigeneraattori, joka perustuu Marlett-algoritmiin ja Cadat-vikasimulaattoriin. Järjestelmä käyttää Cadat-kytkentälistaa ja -mallitusta. Intelligen pystyy käsittelemään tavanomaisten kombinaatiopiirien lisäksi sekvenssipiirejä ja takaisinkytkentäsilmukoita mukaanlukien scan-path-kytkennät. Syöttötietoina ovat kytkentälista, mallit, piirin toimintaa selvittävä ohjaustiedosto, valmiit testiherätteet ja mallien totuustaulut.

Intelligen sallii käyttäjälle mahdollisuuden vaikuttaa generointiin muunmuassa rajoittamalla käytetyn CPU-ajan tietylle tasolle, rajoittamalla perääntymisten määrää, ohjaamalla tietokoneen muistinkäyttöä ja määrittelemällä vikajoukko.



Kuva 3.2 Intelligen. [7]

Intelligen sisältää testattavuusanalyysin, jota käytetään interaktiivisesti ja jossa voidaan dynaamisesti muuttaa piirin kytkentää parhaan mahdollisen testattavuuden hakemiseksi.

Tuloksena koko prosessista saadaan testattavuusanalyysin tulosten lisäksi selvitys kytkennän topologiasta ja käytetyistä totuustauluista, koko generoinnin logtiedosto, raportit vikasimuloinnin osalta, havaittujen ja havaitsemattomien vikojen listat sekä testiherätteet Cadat-muodossa.

Hinta on noin 200 000 USD (-88).

3.2.2 Soveltuvuus ja liitettävyys

Intelligen pohjautuu Cadat-vikasimulaattoriin, joten Micronasin solukirjastot olisi mallitettava uudelleen. Lisäksi Intelligenin tehokas käyttö vaatii Cadat-vikasimulaattorin tuntemusta, jolloin suunnittelijat joutuisivat oppimaan uuden suunnitteluympäristön.

Perusidealtaan Intelligen olisi soveltuva, koska se pystyy tuottamaan testin kombinaatiologiikalle, sekvenssilogiikalle, niiden yhdistelmälle tai scan-tekniikkaa sisältävälle kytkennälle.

Ohjelmistosta on saatavana sekä Unix- että VMS-versio, joten liittäminen Micronasin suunnittelujärjestelmään on mahdollista.

Cadatin primitiivivalikoima on erittäin laaja ja kattaa hyvin mikropiirisuunnittelun tarpeet.

3.3 HILO-3

3.3.1 Yleistä [12]

Hilo-3 on logiikka- ja vikasimulaattori, johon on yhdistetty testivektorigeneraattori. Sen testigeneraattori perustuu kriittisen polun algoritmiin ja sillä on rajoitettu kyky käsitellä sekvenssilogiikkaa. Käyttäjä voi suunnata heräteratkaisua ennakkoon. Lisäksi käyttäjä voi määritellä maksimin CPU-ajalle, iterointikierrosten määrälle sekä tavoitekattavuuden.

Järjestelmästä on liityntä erillisen jälkikäsittelyohjelmiston kautta muutamaan Genrad-testeriin.

Hilo-3 on saatavilla yleisimpiin käyttöjärjestelmiin.

3.3.2 Soveltuvuus ja liitettävyys

Hilo-3:n soveltuvuus Micronasin käyttöön on heikohko johtuen pääasiassa sen puuttellisista kyvyistä käsitellä sekvenssilogiikkaa. Lisäksi järjestelmä käsittää aina myös logiikkasimulaattorin, jolle Micronasin tapauksessa ei olisi käyttöä. Järjestelmää pidetään myös hankalakäyttöisenä.

3.4 AIDA

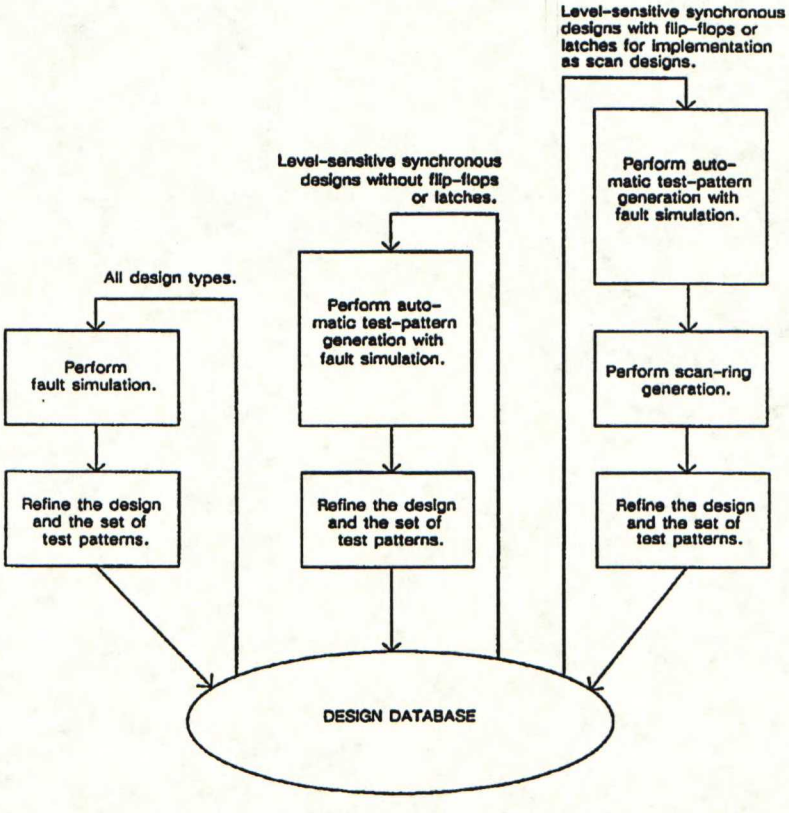
3.4.1 Yleistä [12]

Aidan vektorigeneraattori on osa Aidan suunnittelujärjestelmää. Se perustuu eräänlaiseen D-algoritmin muunnokseen ja edellyttää täydellistä scan-suunnittelua. Vektorigeneraattori käyttää Aidan logiikka- ja vikasimulaattoria ja Aidan kytkentälistaa. Systeemiin kuuluu scan-ketjun generoiva ohjelmisto, joka muuttaa tavalliset kiikut scan-kiikuiksi ja lisää tarvittavan ohjauslogiikan. Järjestelmään voi liittää erillisen Cosim-kiihdyttimen, jolla toimintaa voidaan huomattavasti nopeuttaa.

Ohjelmisto on saavavissa seuraaviin laiteympäristöihin: Apollo, IBM Mainframe ja Sun.

Järjestelmästä on suora liityntä Sentry 20 -testeriin.

Hinta noin 110 000 USD (-88).



Kuva 3.3 Aida:n testintuottostrategia

3.4.2 Soveltuvuus ja liitettävyys

Koska AIDA on kokonaisvaltainen suunnittelujärjestelmä, jonka osana on testivektorigeneraattori, ei se sovellu Micronasin tarpeeseen hankkia pelkkä testivektorigeneraattori. Lisäksi se ei pysty tuottamaan testiä, jos kytkentä sisältää sekvenssipiirejä, joiden ei haluta kuuluvan scan-ketjuun.

3.5 TESTSCAN

3.5.1 Yleistä [12]

Testscanin on kehittänyt Gateway Design Automation Corporation. Se on tarkoitettu scan-rakenteisen kytkennän testivektoreiden kehittämiseen. Testscan käyttää Podem-algoritmia lisättynä alkuun generoiduilla satunnaisvektoreilla. Lisäksi Testscan sisältää testivektoreiden tiivistämiseen tarkoitetun ohjelmiston.

Testscan on saatavana hyvin moniin laitteistoihin kuten IBM PC/AT ja Micro Vax.

Hinta versiosta riippuen on 75 000 - 150 000 USD (-88).

3.5.2 Soveltuvuus ja liitettävyys

Koska Testscan vaatii täyden scan-rakenteen, se ei sovellu Micronasin tämänhetkisiin tarpeisiin.

3.6 LASAR 6

3.6.1 Yleistä [12]

Lasar 6 on Teradyne Inc:in kehittämä ohjelmistopaketti, joka sisältää testivektorigeneraattorin lisäksi vika- ja logiikkasimulaattorin sekä ajoitustarkistimen. Testivektorigeneraattori pystyy tuottamaan täysin kattavan testin scan-rakenteiselle piirille tai kombinaatiologiikalle. Tavanomaisen sekvenssilogiikan suurin syvyys on noin 100 vektoria ja kaikenkaikkiaan ohjelmisto pystyy käsittelemään vain alle 4000 portin kytkentöjä. Generointiohjelmisto perustuu kriittisen polun algoritmiin.

Ohjelmisto on saatavissa sekä VMS-laitteisiin että Unix-ympäristöön.

Koko ohjelmiston hinta on noin 250 000 USD (-88).

3.6.2 Soveltuvuus ja liitettävyys

Lasar 6 ei ole kovin soveltuva Micronasin tarpeisiin, koska se sisältää vektorigeneraattorin lisäksi simulaattorit. Lisäksi kytkennän koon rajoitus alle 4000 porttiin estää Lasar 6:n käytön lähes kaikissa projekteissa.

3.7 TANTEST

3.7.1 Yleistä [12]

Tantest on Tangent Systems Corporationin kehittämä testingenerointiohjelmisto, joka vaatii avukseen Tancell-nimisen sijoitteluohjelmiston. Tantest pystyy tuottamaan testin scan-suunnitellulle kytkennälle ja se pystyy myös itse muuttamaan normaalin sekvenssilogiikan scan-suunnitelluksi logiikaksi. Tantest perustuu kriitisen polun algoritmiin, ja se sisältää myös testivektoreiden

tiivistysohjelman. Käyttäjä pystyy vaikuttamaan generointiajoon asettamalla suurimman sallitun CPU-ajan ja halutun kattavuuden. Tantest on saatavana VMS - ympäristöön. Hinta on Tancell ja Tantest yhdessä noin 250 000 USD (-88).

3.7.2 Soveltuvuus ja liitettävyys

Tantest ei scan-suunnittelun pakollisuuden vuoksi sovellu Micronasin tarpeisiin. Lisäksi ohjelmisto edellyttää Tancellin käyttöä, joka puolestaan toisi turhaan ylimääräisen työkalun järjestelmään.

3.8 TEGAS 5

3.8.1 Yleistä [12]

Tegas 5 on Calma Co:n testivektorigeneraattori, joka generoi testin Calman oman piirikuvauksen pohjalta. Se perustuu laajennettuun D-algoritmiin, jolloin sillä on melko rajoitettu kyky käsitellä sekvenssilogiikkaa. Ohjelmisto sisältää testivektorigeneraattorin lisäksi testattavuusanalyysin. Käyttäjä voi vaikuttaa generointiajoon asettamalla suurimman sallitun CPU-ajan, suurimman sallitun vektorien lukumäärän ja vikakohtaisten iteraatiokierrosten maksimimäärän. Tegas 5 on saatavana Data General- IBM- ja Vax-laitteistoihin.

3.8.2 Soveltuvuus ja liitettävyys

Tegas 5 ei sovellu Micronasin tarpeisiin, koska sen kyky käsitellä sekvenssilogiikkaa on liian rajoitettu.

3.9 ATTE

3.9.1 Yleistä [10, s.4]

Atte on Nokia Tutkimuskeskuksessa kehitetty testivektorigeneraattori, joka pystyy tuottamaan täysin kattavan testin kaikille havaittaville yksittäisille stuck-at-vioille sekä kombinaatiologiikalle että scan-suunnitteluille. Atte hyväksyy myös osittaisen scan-suunnittelun, mutta testin kattavuus saattaa silloin jäädä alhaisemmaksi. Atte perustuu Nokian omaan algoritmiin ja käyttää lisäksi satunnaisvektoreita. Atte on saatavana Mentor Graphics Idea -järjestelmään.

3.9.2 Soveltuvuus ja liitettävyys

Atte ei täytä kaikilta osin Micronasin tarpeita. Koska Atte ei tue täysin scan-rakenteetonta suunnittelua, ei sillä pystytä tuottamaan testiä kuin osaan Micronasin suunnitteluista. Lisäksi Attea ei ole saatavana Vax-järjestelmään. Atte-generaattorin jatkokehitystyö on aloitettu ja tarkoituksena on kehittää yleiskäyttöisempi ohjelmisto.

3.10 YHTEENVETO

Kuten edelläesitetystä voidaan todeta, ei Micronasin tarpeita vastaavia järjestelmiä ole kuin kaksi: Intelligen ja Nextgen, jotka valittiin jäljempänä esitettäviin koeajoihin. Kaikkien muiden puutteena on joko ehdoton scan-suunnittelun vaatimus tai puutteellinen kyky käsitellä sekvenssilogiikkaa. Useat järjestelmät myös vaativat toimiakseen kokonaisen suunnittelujärjestelmän simulaattoreineen, kun tässä projektissa puolestaan päämääränä on löytää erillinen toimiva testivektorigeneraattori.

4 ATPG-OHJELMIEN KOEAJOT

4.1 YLEISTÄ

Koeajettaviksi valittiin kaksi kaupallista tuotetta eli HHB Systemsin Intelligen ja Zycad Corp:n Nextgen. Koeajot suoritettiin kahdessa vaiheessa Lontoossa kummankin toimittajan tiloissa. Koska laiteympäristö oli kummassakin tapauksessa Micronasin laiteympäristöstä poikkeava, on lopullisissa vertailutuloksissa pyritty huomioimaan esimerkiksi eritehoisten keskusyksikköjen aihettamat erot prosessointiajoissa. Koeajojen ensimmäisessä vaiheessa saatiin kohtuullisen hyvä perustuntuma järjestelmien ominaisuuksiin. Toisessa vaiheessa perehdyttiin syvällisemmin alustavasti valitun järjestelmän erityisominaisuuksiin.

Koeajojen lisäksi tehtiin tehtiin tarkka kartoitus toimittajien taustasta, tukitoiminnoista, muista käyttäjistä ynnä muista vastaavista asioista.

4.2 TESTIKYTKENNÄT

4.2.1 BM1

BM1-nimellä tunnettu kytkentä on Testa-projektin koekytkentä n:ro 1, joka on noin 300 portin synkroninen sekvenssipiiri. Sen kytkentäkaaviot on esitetty liitteessä 1. BM1 on tarkoitettu lähinnä selvittämään testigeneraattoreiden prosessointinopeutta. BM1 muodostuu kolmesta lohkosta, jotka muodostuvat tilakoneesta, kombinaatiologiikalla lisätystä siirtorekisteristä ja puhtaasta kombinaatiologiikasta.

4.2.2 BM2

BM2-nimellä tunnettu kytkentä on erään tuotteen digitaaliosa, joka on noin 2000 portin asynkroninen sekvenssipiiri. Sen kytkentäkaaviot on esitetty liitteessä 2. BM2 on tarkoitettu selvittämään järjestelmien kykyä löytämään ongelmapaikkoja ja arvioimaan testattavuutta. BM2 on testattavuuden kannalta erittäin hankalasti suunniteltu kytkentä, joka sisältää pääasiassa erilaisia laskureista ja kombinaatiologiikasta muodostettuja tilakoneita.

4.2.3 Pilot

Pilot-nimellä tunnettu kytkentä on myös erään tuotteen digitaaliosa. Se on noin 3000 portin osittain synkroninen ja osittain asynkroninen kytkentä, jolle on manuaalisesti tehty testivektorit aiemmin. Pilot muodostuu hyvin erilaisista lohkoista, kuten asynkronisella jakajalla toteutettu kellogeneraattori, erilaiset osoitedekooderit, tilakoneet ja digitaalinen vaihelukko. Pilot-kytkentää käytettiin koeajojen toisessa vaiheessa lähinnä järjestelmien erityispiirteiden selvittämiseen, jolloin generoitiin testejä pääasiassa alilohkoille.

4.3 TULOKSET

BM1- ja BM2-kytkentöjen avulla saatiin muodostetuksi selvä ero tutkittujen järjestelmien välillä. Vertailuominaisuuksina käytettiin prosessointitehoa, testingeneroinnissa saavutettua kattavuutta, käyttäjäliitynnän laatua, testattavuusanalyysin tasoa, Euroopassa tarjottavan tuen tasoa sekä hintaa.

BM1	Nextgen	Intelligen
TESTATTAVUUSANALYYSI		
suoritusaika	36s	8s (32s) [1]
ei-testattavat solmut	22	19
analyysin taso	hyvä	hyvä
TESTIVEKTORIGENEROINTI		
vikojen kokonaismäärä	704	681 (365) [2]
kattavuus-%	92.76	96.43
vektorien lukumäärä	423	405
host-koneen CPU-aika	9min 13s	62s (4min 8s) [1]
kiihdytinaika [3]	31s	-
raporttien taso	hyvä	hyvä
MUUT SEIKAT		
Tuen taso	tyydyttävä	hyvä
Järjestelmän kokonaishinta	410.000 USD	200.000 USD

1) CPU-aika on suluissa muunnettuna 1 MIPS-tehoisen koneen suoritusajaksi.

2) Vikojen määrä suluissa on todellinen prosessoitavana ollut vikamäärä ohjelman automaattisesti tekemän redundanttisten vikojen poiston jälkeen.

3) Nextgen edellyttää simulointikiihdyttimen käyttöä.

BM2:lle tehdyissä koeajoissa todettiin Intelligenin testattavuusanalyysi huomattavasti paremmaksi. Intelligen pystyi selkeästi osoittamaan ongelmapaikat ja suoritettujen muutosten jälkeen testingenerointi onnistui hyvin kaikille lohkoille kattavuuden asettuessa 80% - 100 %:n tasolle.

Nextgen ei pystynyt tuottamaan järkevää testattavuusanalyysiä BM2:lle.

4.4 Ohjelmiston valinta

Koeajoissa saatujen kokemusten ja tietojen pohjalta todettiin, että Intelligen on ainut varteenotettava kaupallinen Micronasille soveltuva testingenerointiohjelmisto. Intelligen on samantehoisessa keskusyksikössä testingeneroinnissa yli kaksi kertaa nopeampi kuin Nextgen ja suunnilleen yhtä nopea testattavuusanalyysissä. Koska testingenerointiin kuluu kuitenkin useita kertaluokkia enemmän aikaa kuin testattavuusanalyysiin, on yli kaksinkertainen nopeusero siinä selvä etu. Intelligen tukee erittäin tehokkaasti osittaista scan-rakennetta, joka puolestaan on Micronas Oy:n tarpeisiin erityisen hyvin soveltuva menetelmä testattavuuden paratamiseksi. Nextgen vaikutti olevan hieman keskeneräinen, koska esimerkiksi testattavuusanalyysi ei onnistunut BM2:lle lainkaan. BM1:lle tehdyssä testingeneroinnissa Intelligen saavutti paremman kattavuuden eron jäädessä tosin pieneksi. Intelligenin lisäetuina pidettiin sen huomattavasti alhaisempaa hintaa ja selvästi paremman tuntuista tukea Euroopassa.

5 OHJELMISTON KÄYTTÖÖNOTTO

5.1 SIJOITTAMINEN OSAKSI SUUNNITTELUJÄRJESTELMÄÄ

Intelligeniä varten Micronas Oy hankki aiempia tietokoneita tehokkaamman Vax Station 3100:n, jonka prosessointiteho on noin 4 MIPS. Ohjelmistoa voidaan lisäksi käyttää muissa 1 MIPS:n tehoisissa VAX-keskusyksiköissä. Liitteessä 2 on esitettynä Micronas Oy:n suunnittelujärjestelmän laitteisto, jossa nimellä MASVXK on merkittynä Vax Station 3100. Ohjelmisto on asennettu siten, että varsinaiset ohjelmat ovat omalla levyllään, mallikirjastot ja muut yleiset tiedostot omalla levyllään ja projekteihin liittyvät tiedostot omalla levyllään. Kaikki Vax Stationit ja Micro Vaxit on liitetty toisiinsa Ethernetillä ja muodostavat Vax Clusterin, jossa kaikki keskusyksiköt voivat käyttää yhteisiä tiedostoja.

Työkaluna ohjelmisto sijoittuu muiden logiikkasuunnittelun työkalujen joukkoon liitteen 3 mukaisesti. Intelligen käyttää Silos2:n tuottamaa tai myöskin CT-ohjelmistosta saatua tai Mentor-Cadat-muunnosohjelmasta saatavaa kytkentälistaa. Prosessoinnin tuloksena se tuottaa testin, joka siirretään eri vaiheiden kautta Tiosat-ohjelmiston avulla testerille.

5.2 KIRJASTOMALLIT JA KYTKENTÄLISTAT

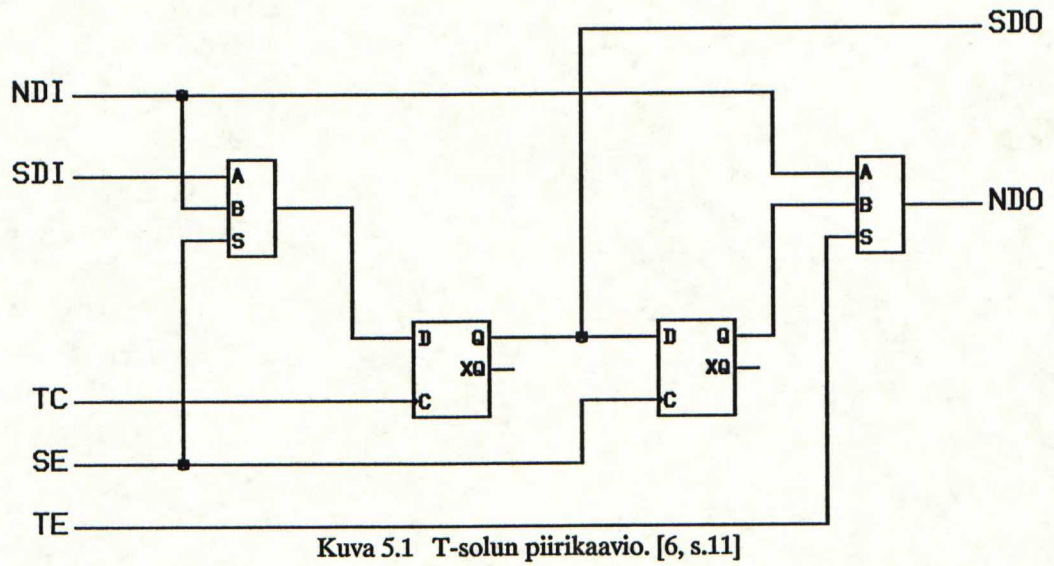
Intelligen vaatii toimiakseen Cadat-muodossa olevan kytkentälistan sekä Cadat-muotoiset mallit solukirjastolle. Sopimukseen HHB Systemsin kanssa kuuluu Micronas Oy:n kahden solukirjaston mallittaminen HHB Systemsin toimesta. Micronas Oy:n tehtäväksi jäi mallituksen tarkistaminen. Kirjastomallien lisäksi Intelligen tarvitsee eräänlaiset totuustaulut jokaiselle käytössä olevalle primitiiville. Totuustaulujen generointia varten Intelligenissä on erillinen ohjelmisto. Kytkentälistan tuottamiseen Micronas Oy:n käyttämästä piirikaavioeditorista on olemassa kaksi vaihtoehtoa. Kytkentälista voidaan tuottaa joko Tocadat-nimisellä ohjelmalla piirikaavioeditorin tuottamasta yleiskäyttöisestä kytkentälistasta tai käyttämällä Silos2-logiikka- ja vikasimulaattorin yhteydessä olevaa Silos-Cadat-muunnosohjelmaa. Silos-Cadat-muunnosohjelma muuntaa lisäksi Silos-muotoiset herätteet Cadat-muotoon sekä Silos-muotoiset solukirjastomallit Cadat-muotoon.

5.3 T-SOLU JA SEN OHJAUSSOLU

T-solu on HHB Systemsin kehittämä testattavuuden lisäämisrakenne, jolla saadaan helposti lisättyä osittaisen scan-rakenteen avulla kytkennän sisäisten solmujen ohjattavuutta ja havaittavuutta vaikuttamatta piirin varsinaiseen toimintaan juuri lainkaan. Normaalitilassa T-solu ilmenee vain muutaman nanosekunnin viiveenä siinä signaalissa, johon se on kytketty. Intelligeniin kuuluvat apuohjelmat mahdollistavat osittaisen scan-rakenteen käytön lähes automaattisesti.

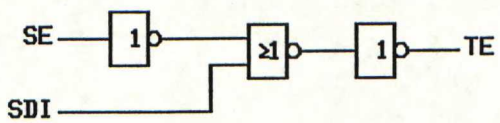
T-solu on loogiselta rakenteeltaan kuvan 5.1 mukainen. Se toimii eräänlaisena scan-soluna sallien normaalissa toimintatilassa signaalin kulun häiriintymättä

suoraan läpi. Testitilassa sillä joko ohjataan sen lähtöön NDO kytkettyä kytkennän tuloa tai sen avulla tutkitaan sen tuloon NDI kytkettyä kytkennän lähtöä.



Kuva 5.1 T-solun piirikaavio. [6, s.11]

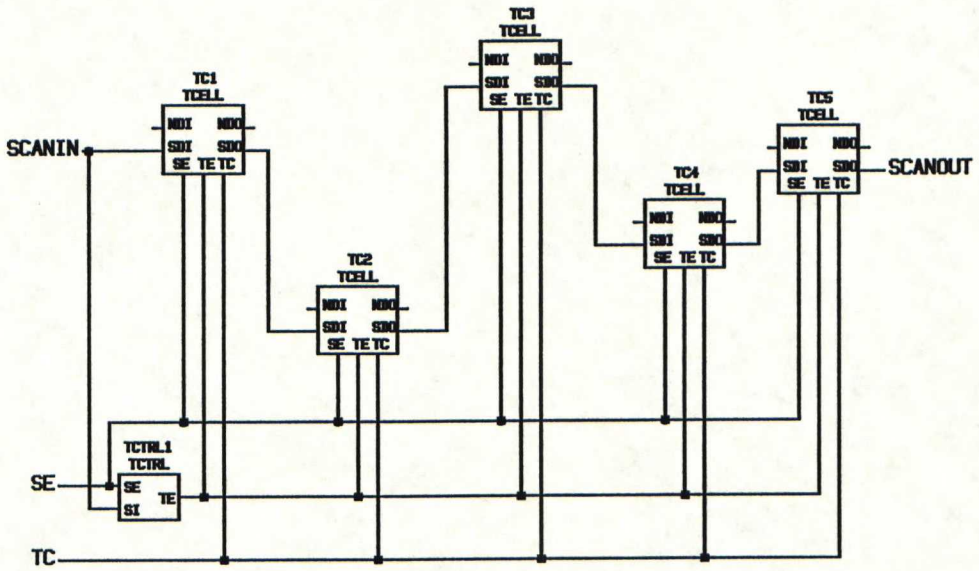
T-soluja ohjaava logiikka on esitetty kuvassa 5.2.



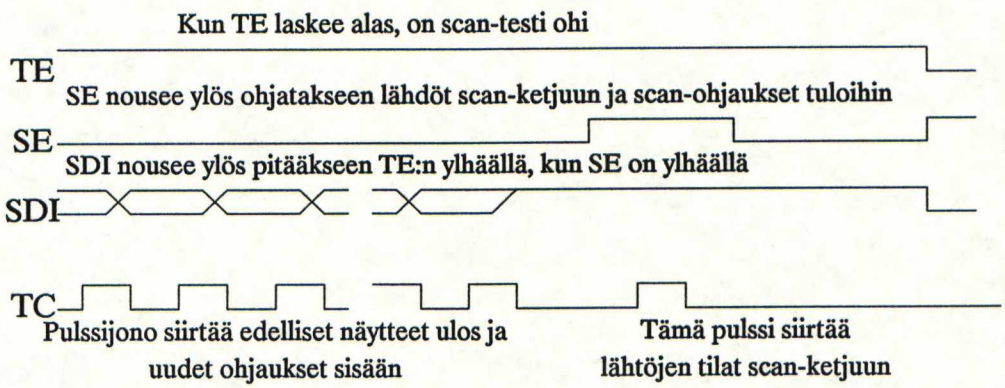
Kuva 5.2 T-solujen ohjauslogiikka. [6, s.12]

T-solut kytketään toisiinsa ketjuksi SDI:n ja SDO:n avulla kuvan 5.3 mukaisesti, jolloin ohjaava data tai tutkittava data kulkee piirin läpi sarjamuodossa.

Kuvassa 5.4 on esitettynä T-solun ohjaussignaalit ja toiminnan eri vaiheet.



Kuva 5.3 T-solut kytkettynä ketjuksi. [6, s.12]



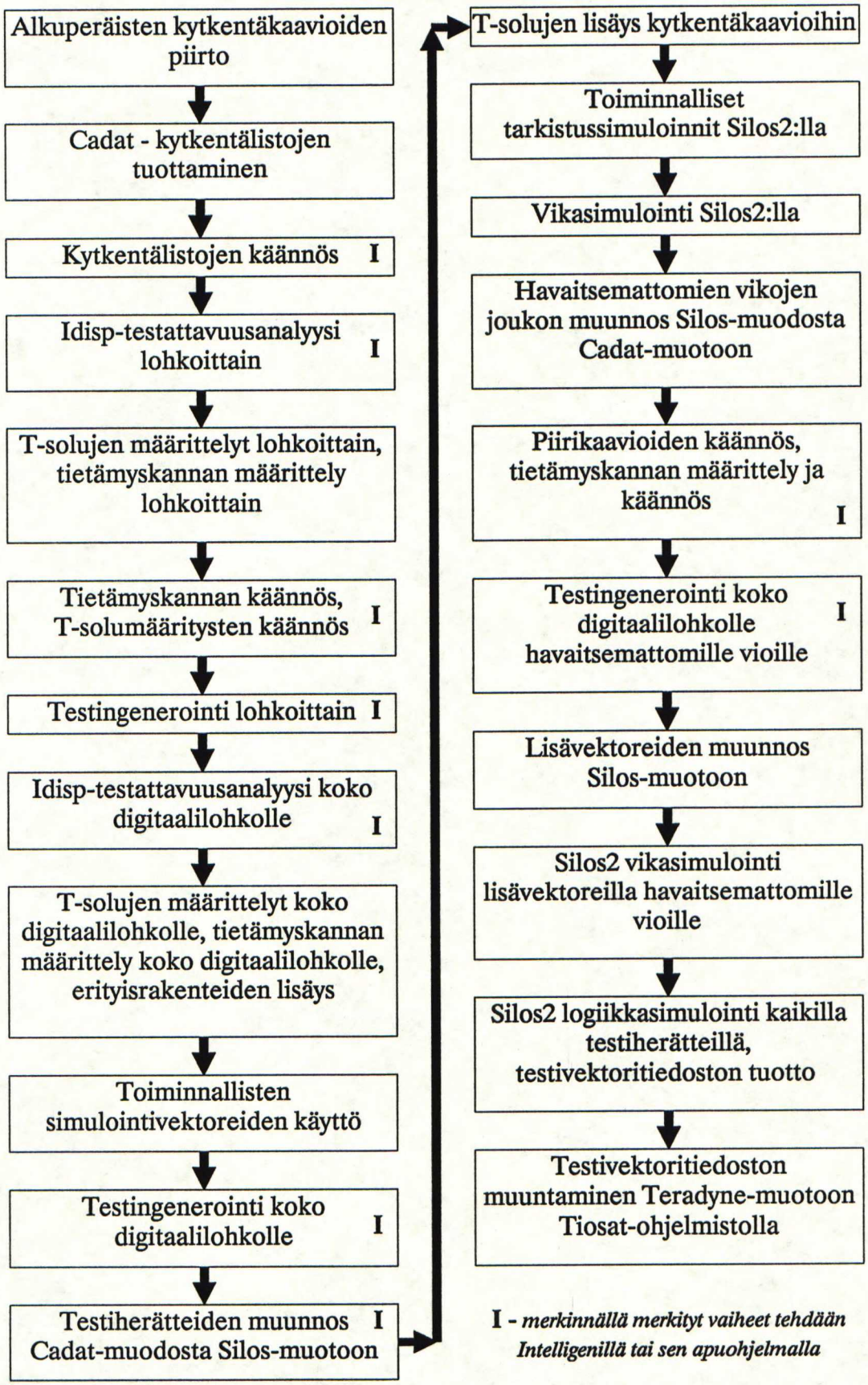
Kuva 5.4 T-solun ohjaussignaalit. [6, s.13]

5.4 KÄYTTÖRUTIINIT JA SOVELLUSOHJEISTO

Tässä luvussa esitetään tämän diplomityön osana suunniteltu ohjeisto, joka opastaa suunnittelijaa testivektoreiden generoinnissa vaihe vaiheelta. Esitettävä ohjeisto ei ole täydellinen esimerkiksi Intelligenin tai muiden ohjelmistojen varsinaisten käyttöohjeiden osalta, vaan tämä ohjeisto ohjaa suunnittelijaa päävaiheiden osalta.

Perusideana on tuottaa tuotteen digitaalilohkolle testivektorit Intelligen-ohjelmistolla ja varsinaisesti tarkistaa testin kattavuus Silos2:lla, joka on Micronas Oy:n virallinen logiikkasimulaattori.

Seuraavalla sivulla esitetyssä kuvassa 5.5 on koko suunnittelutyön kulku kytkentäkaaviosta valmiiseen testiohjelmaan asti. Tämän vuokaavion tarkoituksena on selventää lukijalle prosessin kokonaiskuvaa.



Kuva 5.5 Suunnittelutyön kaikki vaiheet kytkentäkaaviosta testiohjelmaan.

5.4.1 Cadat-kytkentälistan tuottaminen

Cadat-tyyppinen kytkentälista voidaan tuottaa Micronas Oy:n käyttämästä piirikaavioeditorista kahdella eri menetelmällä : Tocadat-ohjelmalla tai Silos2:n Silos-Cadat-muunnosfunktiolla.

5.4.1.1 Silos2:n muunnosohjelma

Silos2:n muunnosfunktio tuottaa Cadat-kytkentälistan tietyin puuttein. Funktio määrittelee Nofaults-kenttään kaikki kytkennän komponentit, jolloin Intelligen ei aseta vikoja niiden lähtöihin tai tuloihin. Nofaults-kenttä on siis editoitava tyhjäksi. Muunnosfunktio jättää Externals-kentän tyhjäksi, joten siihen on lisättävä kaikki primääritulot ja -lähdöt oikeassa muodossa. Lisäksi Intelligen tarvitsee kaikille kaksisuuntaisille primäärisignaaleille kaksi nimeä ymmärtääkseen ne sekä tuloiksi että lähdöiksi. Kaksisuuntaiset signaalit on siis nimettävä Connections-kentässä uudelleen kahdella eri nimellä, jotka merkitään vastaavasti Externals-listaan.

Esimerkki:

```
CONNECTIONS $
$    BUS5  ← alkuperäinen signaalinimi
                U0154.3
+               BUS5IN        ← muutetut signaalinimet
+               BUS5OUT $ 2

EXTERNALS $
     IN5 (IN) $
     IN6 (IN) $                    ← lisätyt signaalinimet varustettuna
     OUT4 (OUT) $                        IN/OUT-merkinnällä
     BUS5IN (IN) $
     BUS5OUT (OUT) $

NOFAULTS $                         ← tyhjennetty Nofaults-kenttä
END $
```

5.4.1.2 Tocadat-ohjelma

Tocadat tuottaa lähes valmiin kytkentälistan Cadat-muotoisena. Ainoana puutteena mainittakoon kaksisuuntaiset primäärisignaalit, joiden uudelleen nimeäminen pitää suorittaa samoin kuin Silos2:n muunnosohjelman tapauksessa. Tocadat vaatii piirikaavioeditorin tuottaman kytkentälistan esiprosessoinnin Tolayout/nopwr-ohjelmalla.

5.4.2 Kytkentälistan käännös

Käännös suoritetaan jokaiselle lohkolle erikseen valitsemalla Intelligenin valikosta toiminnot 'Circuit Processing' ja 'Data Base Generation'. Tässä vaiheessa ei tarvita muita toimintoja. Tarvittaessa voidaan kuitenkin jo nyt tehdä määrittelyjä tietämyskantaan, jolloin myös ne pitää kääntää 'Compile Knowledge Base'-toiminnolla. Käännöksen yhteydessä määritellään myös käytettävät mallikirjastot.

5.4.3 Testattavuusanalyysi lohkoittain

Kytkentälistan käännöksen jälkeen suoritetaan testattavuusanalyysi jokaiselle lohkolle erikseen Idisp-ohjelmalla. Ohjelman ajon aikana voidaan interaktiivisesti tutkia kytkennän testattavuutta joko numeeristen tunnuslukujen tai graafisten esitysten avulla. Lisäksi voidaan kokeiluluontoisesti selvittää, miten kytkennän testattavuuteen vaikuttaisi, jos jokin solmuista muutettaisiin primääriseksi tuloksi ja lähdöksi, eli jos siihen asetettaisiin T-solu. Nyrkkisääntönä voidaan pitää tarkoituksena saada tarvittaessa erilaisilla muutoksilla vaikeimmin ohjattavan solmun ohjattavuusarvot alle 3000:n. Idisp-ohjelman avulla voidaan myös tutkia keskeytyneen testingeneroinnin aikana syntyneitä vektoreita, Intelligenin sisäiseen primitiivimallitukseen liittyviä totuustauluja ja kytkennän topologiaa.

Testattavuusanalyysi ja testingenerointi lohkoittain suoritetaan ylimmän tason testingeneroinnin onnistumisen varmistamiseksi. Tuotettua testiä ei voi käyttää, koska alilohkon primääritulot ja -lähdöt eivät ole useimmissa tapauksissa suoraan käytettävissä. Intelligen ei myöskään pysty käyttämään alemman lohkon testiä hyväkseen ylimmän tason testingeneroinnissa. Jos piirin testattavuus on varmuudella riittävän hyvä, ei lohkotason testattavuusanalyysiä eikä testingenerointia tarvitse suorittaa.

5.4.4 T-solujen ja tietämyskannan määrittely lohkoittain

Testattavuusanalyysin avulla saadaan selville mahdollisesti tarvittavien T-solujen otollisimmat paikat. T-solujen paikat kerrotaan Intelligenille erityisessä tiedostossa, joka on tyypiltään .TLF, ja jossa on yksikertaisesti lueteltuna ne solmunimet, joihin T-solu halutaan asettaa.

Esimerkki .TLF-tiedostosta:

```
U0134(U1.Q)
U0122(U1.QN)
U0178(U1.Y)
```

Testattavuusanalyysi ja yhtälailla tavanomainen kytkennän tarkastelu antaa usein tietoa siitä, miten esimerkiksi jokin kytkennän osa saadaan parhaiten havaittavaksi tai ohjattavaksi. Kysymyksessä saattaa olla esimerkiksi joidenkin signaalien pakko-ohjaus tiettyyn tilaan tai ohjaus tiettyyn tilaan testin ensimmäisellä vektorilla. Myöskin on mahdollista, että esimerkiksi jokin tilako-

ne ei voi koskaan saavuttaa tiettyä tilaa, jolloin on hyvä määritellä sellainen tila Intelligenille kielletyksi tilaksi, ettei se turhaan käytä aikaa sen tilan saavuttamiseksi. Tämän kaltaiset asiat saadaan välitettyä Intelligenille tietämyskantaan tehtävien määrittelyjen avulla. Määrittelyjä tehdessä kannattaa olla erityisen huolellinen, jotta tavoiteltu etu saadaan varmasti ja oikealla tavalla käyttöön.

5.4.5 Tietämyskannan ja T-solumääritysten käännös

Ennen T-solumääritysten käännöstä pitää kytkentälistaan lisätä Parts-kenttään sanat 'Testio Testio $', jossa ensimmäinen 'Testio' vastaa komponentin sijaintitunnusta ja toinen on sen nimi. '$' päättää rivin. Lisäys on tehtävä testattavuusanalyysin jälkeen, koska Testio-komponentti kytkennän käännöksessä heikentää keinotekoisesti sekventiaalisten Cadat-primitiivien ohjattavuus- ja havaittavuusominaisuuksia, ja siten analyysi antaisi todellisuudesta poikkeavia tietoja. Lisäyksen varsinainen tarkoitus on tuottaa T-solumääritysten kääntäjälle eräänlainen T-soluvarasto, josta se ottaa T-solumalleja tarvittaessa. Sen vuoksi varsinaiseen kirjastoon käännetyn Testio-mallin koko pitää olla vähintään käytettävien T-solujen määrä. Lisäyksen jälkeen on luonnollisesti käännettävä kytkentälista uudelleen. Varsinainen T-solumääritysten käännös tapahtuu Tlf-ohjelmalla, joka muuttaa binäärimuotoista kytkennän tietokantaa siten, että Intelligen käyttää T-solujen paikoiksi määriteltyjä solmuja primäärituloina ja -lähtöinä.

Tietämyskanta käännetään valitsemalla Intelligenin valikosta toiminto 'Knowledge Base Compile'.

5.4.6 Testingenerointi lohkoittain

Varsinainen testingenerointi käynnistetään valikosta valitsemalla 'Test Generation'. Sitä ennen on kuitenkin määriteltävä joukko asioita, jotta generointi tuottaisi mahdollisimman hyvän tuloksen. Nämä lisämääritteet tehdään Intelligenin valikon toisella sivulla, jonne siirrytään automaattisesti, jos määrittelyjä ei ole tehty lainkaan. Perääntymismäärän maksimi on pienten lohkojen tapauksessa usein sopiva oletusarvossaan 200. Samoin vikakohtaisen CPU-ajan maksimi 240 s on useimmiten sopiva. Testingeneroinnin nopeuttamiseksi on syytä valita valitsin '-j', koska se määrittää ohjelmiston pitämään testivektorit koko ajan keskusmuistissa eikä levymuistissa. Generoinnin aikana näytössä on nähtävänä tietoa prosessin etenemisestä, ja siitä voidaan helposti päätellä, sujuuko prosessi oikein. Lohkotasoisen testingeneroinnin tarkoituksena on paljastaa mahdolliset sellaiset ongelmarakenteet ja -tilanteet, joita testattavuusanalyysi ei paljasta. Syntyneellä testillä ei ole varsinaista käyttöä, vaan tieto siitä, että riittävän kattava testi on tuotettavissa, antaa olettaa, että testi pystyttäneen tuottamaan myös, kun tämä kytkentä on osana suurempaa kytkentää.

5.4.7 Testattavuusanalyysi koko digitaalilohkolle

Kun testattavuus on lohkotasolla saatu asianmukaiseen kuntoon, voidaan siirtyä seuraavalle tasolle hierarkiassa (tässä esityksessä ylimmälle tasolle).

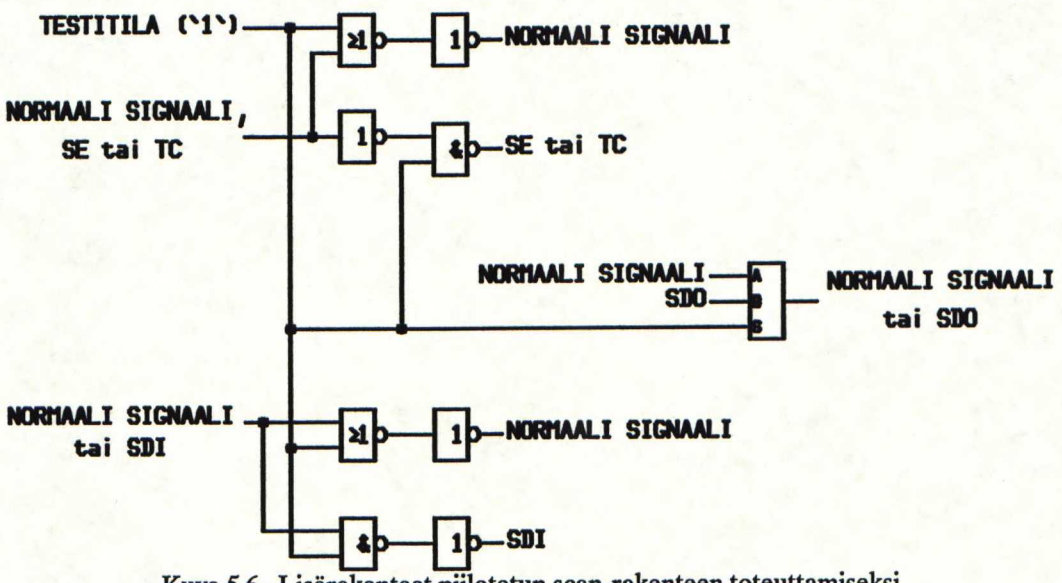
Testattavuusanalyysi tehdään samalla tavoin kuin lohkotasolla. Kun T-solujen paikkoja kokeellisesti määritellään, on ensin muistettava määritellä T-solut niihin paikkoihin, joihin ne määriteltiin lohkotasolla. Sen jälkeen määritellään lisää T-soluja tarpeen mukaan. Koko digitaalilohkon testattavuusanalyysin tavoitteena voidaan pitää heikoimmin ohjattavan solmun ohjattavuuslukujen saamista alle 20000:n.

5.4.8 T-solujen ja tietämyskannan määrittely koko digitaalilohkolle

Nämä määrittelyt tehdään samoin kuin lohkotasolla.

5.4.9 Testirakenteiden lisäys piilotetun scan-rakenteen tapauksessa

Piilotetulla scan-rakenteella tarkoitetaan menetelmää, jossa scan-ketjun ohjaamiseen tarvittavat signaalit on multipleksattu kolmen piirin tulon ja yhden lähdön kanssa. Piiriin tarvitaan yksi ylimääräinen tulo, jolla piiri asetetaan testitilaan. Testitilassa multipleksatut kolme tuloa toimivat vain scan-ketjua ohjaavina tuloina ja multipleksattu lähtö vain scan-ketjun lähtönä. Toiminta saadaan aikaiseksi kuvan 5.6 esittämillä ratkaisuilla. Nämä testirakenteet on syytä lisätä tässä vaiheessa, jotta niihin asetettavat viat ovat huomioitavina jo nyt. Suurin osa näiden rakenteiden vioista jää paljastumatta ensimmäisellä testingeneroinnilla, koska testitilaan ohjaava tulo on määriteltävä tietämyskannassa kiinteästi aktiiviseen tilaan. Oleellista kuitenkin on, että paljastumatta jäävät viat on huomioitu, jolloin niille tullaan tuottamaan testi myöhemmin.



Kuva 5.6 Lisärakenteet piilotetun scan-rakenteen toteuttamiseksi

5.4.10 Toiminnallisten simulointivektoreiden käyttö apuna

Ennen generoinnin aloittamista on usein hyödyllistä selvittää, mitkä viat tulevat havaituiksi pelkillä toiminnallisilla simulointiherätteillä. Usein kattavuus asettuu 50 - 70%:iin. Näin tapahtuva vikajoukon esikarsinta suoritetaan kääntämällä herätteet Cadat-logiikka- ja vikasimulaattorille sopivaan muotoon Silos2:lla

ja suorittamalla vikasimulointi Cadatilla, josta saadaan havaitsemattomien vikojen lista. Vikasimulointi voidaan tehdä myös Silos2:lla, mutta vikajoukon muuntaminen Silos-muodosta Cadat- eli Intelligen-muotoon vaatii editointia manuaalisesti ja on siten virhearka työvaihe. Vikasimulointi Silos2:lla on toisaalta pakollinen, jotta saadaan selville havaitsemattomien vikojen joukko seuraavaa vikasimulointia varten.

Herätteiden muunnos Cadat-muotoon tapahtuu Silos2:n Silos-Cadat-muunnosohjelmalla.

On huomattava, että vikasimulointi toiminnallisilla herätteillä pitää tehdä täysin alkuperäistä kytkentää käyttäen, jotta herätteet toimivat oikein ja, ettei toisaalta vikajoukkoon tule sellaisia vikoja, joita ei testirakenteettomassa kytkennässä ole lainkaan.

5.4.11 Testingenerointi koko digitaalilohkolle

5.4.11.1 Karsimaton vikajoukko

Jos yhdellekään vialle ei ole vielä testiä tai testingenerointia niille pidetään helppona, suoritetaan testingenerointi normaalin tavan mukaisesti Intelligenin valikon kautta toimien. Testingenerointia valmistellessa on asetettava valikon eri sivuilla olevat valitsimet ja määritteet oikein. Hyvänä perusasetuksena esimerkiksi suurimmaksi sallituksi perääntymismääräksi voidaan pitää 2000:a. Vastaavasti sopiva suurin sallittu CPU-aika vikaa kohden on 1200 s. Mainitut arvot on todettu sopiviksi muutaman tuhannen portin kokoiselle kytkennälle, jossa on käytetty osittaista scan-rakennetta. On kuitenkin syytä muistaa, että muunmuassa kytkennän rakenne vaikuttaa voimakkaasti esimerkiksi tarvittavaan CPU-aikaan.

Testingeneroinnin aikana on syytä tarkkailla prosessin etenemistä, jotta mahdolliset ongelmat havaitaan, ennen kuin on kulutettu runsaasti aikaa turhaan. Kohtuullisen hyvä indikaattori testingeneroinnin onnistumiselle on saavutetun kattavuuden lisäksi onnistuneiden ja yritettyjen testingenerointien suhde, joka on näkyvissä testingeneroinnin aikana näytössä.

5.4.11.2 Karsittu vikajoukko

Esimerkiksi toiminnallisia herätteitä käyttäen karsitulle vikajoukolle suoritettava testingenerointi pitää käynnistää valikon sijasta rivikomennolla. Komennon muoto on seuraava:

IGEN "nnnyS" KIRJ1,KIRJ2,TOT1,TOT2 KYTKENTA "-o200,240 %100 -c -j"

Komennon parametrien merkitys ilmenee tarkemmin Intelligen-käyttöohjeesta.

Testingeneroinnin edistymistä on syytä tarkkailla kuten edellisessäkin tapauksessa.

5.4.12 Testiherätteiden muunnos Silos-muotoon

Testingeneroinnin tuloksena saadaan Cadat-muotoinen testiheräte, joka on muunnettava Silos-muotoon Silos2-vikasimulointia varten. Muunnos tapahtuu C2S-ohjelmalla, joka tuottaa uuden .ATP-tiedoston. Se sisältää herätteet Silos-muodossa. Jos kysymyksessä on testingenerointi ilman T-soluja, ohjelma käynnistetään komennolla 'C2S tiedosto'. Jos generointi on tehty käyttäen T-soluja, ohjelma käynnistetään komennolla 'C2S tiedosto -S', jossa lisämäärite '-S' ohjaa ohjelman huomioimaan scan-rakenteen edellyttämät lisävektorit. Scan-rakenteen tapauksessa ohjelma tarvitsee normaalien testingeneroinnin tuottamien tiedostojen lisäksi aiemmin esitetyn .TLF-tiedoston sekä .HOW-tiedoston. Se sisältää rivin, jossa on määriteltyinä scan-rakenteen tarvitsemien ylimääräisten tulojen ja lähdön nimet.

Esimerkki:

SCANIN,SCANOUT,SE,TC;

Piilotetun scan-rakenteen tapauksessa nimiksi on annettava ylimääräisellä testirakenteella varustettujen tulojen oikeat nimet.

Tuotettuja Silos-tyyppisten herätteitä on joissain tapauksissa editoitava vielä käsin, ennenkuin ne ovat käyttökelpoisia. Esimerkiksi testijaksoksi muunnosohjelma olettaa aina 100 ns, joka joudutaan ehkä muuttamaan. Lisäksi piilotetun scan-rakenteen tapauksessa signaalinimiä joudutaan editoimaan siten, että testirakenteellisille tuloille annetaan satunnaiset muut nimet, esimerkiksi DUMMY1 ... Näin toimien Intelligenin näille tuloille tarkoittamat ohjaukset vaihtuvat .HOW tiedoston ansiosta scan-rakenteiden ohjaukseen tarkoitettuihin ohjauksiin. Ylimääräisiä testirakenteita ohjaava aktiivinen lisätulohan pakottaa näihin primärituloihin tulevat signaalit scan-ketjulle.

Silos2-vikasimuloinnissa Intelligenillä tuotettuja herätevektoreita käyttäen on huomioitava, että kaksisuuntaisen primäärisignaalin solun IOL4:n Intelligenmallissa on Intelligenin tarvitsema ylösvetovastus. Koska sitä ei ole vastaavassa Silos-mallissa, on herätteiden osalta määriteltävä Silos2:lle, että herätevektoreissa oleva 'Z' tarkoittaa resistiivistä 1-tilaa. Testerillä tämä ei aiheuta kuitenkaan toimenpiteitä, koska testeriliityntäohjelmisto Tiosatille Silos2:lla tuotettava tiedosto esittää kaksisuuntaisen primäärivaylän signaalit aina normaalivoimakkuuksisina.

5.4.13 T-solujen lisäys kytkentäkaavioihin

Ennenkuin generoituja testivektoreita päästään kokeilemaan lopulliseen scanketjun sisältävään kytkentään, on T-solut lisättävä oikeaan .TLF-tiedoston määrittämään järjestykseen kytkentäkaavioihin. Lisäyksen jälkeen on luonnollisesti tuotettava uusi Silos-muotoinen kytkentälista.

5.4.14 Toiminnan tarkistus Silos2:lla

Ennen vikasimulointia Silos2:lla on syytä tarkistaa, että T-solujen lisäämisessä kytkentäkaavioihin ei ole tehty virheitä. Tarkistus tehdään simuloimalla kytkentä toiminnallisten simulointien herätteillä ja vertaamalla tuloksia.

5.4.15 Vikasimulointi Silos2:lla

Vikasimulointi Silos2:lla antaa selvyyden generoitujen vektoreiden kattavuudesta T-soluilla lisätyssä kytkennässä. Vikasimulointi tehdään siis generoiduilla vektoreilla. Mikäli testingenerointi on tehty toiminnallisten vektoreiden havaitsematta jättämille vioille, on myös tässä simuloinnissa muistettava käyttää rajoitettua vikajoukkoa.

5.4.16 Havaitsemattomien vikojen joukon siirto Intelligeniin

Silos2:lla suoritetusta vikasimuloinnista saadaan havaitsemattomien vikojen lista, joka pitää muuttaa Intelligenin ymmärtämään muotoon. Nykyisellään muunnokseen ei ole automaattista menetelmää, joten muunnos pitää tehdä editoimalla manuaalisesti.

Esimerkki vikajoukon muuntamisesta:

Silos2:

```
.SHIGH
U0110%QDAA

.SLOW
U0123%IN1AAA
```

Intelligen:

```
U0110DAA.4 /1  $
U0123AAA.1 /0  $
```

Muunnettu vikajoukko talletetaan tiedostoon, jonka tyyppi on .FSL.

Tyypillisesti havaitsemattomiksi vioiksi ovat jääneet esimerkiksi piilotetun scan-rakenteen vuoksi lisättyjen erityisrakenteiden viat sekä muutamat itse scan-rakenteen viat.

5.4.17 Piirikaavion käännös, tietämyskannan määrittely ja käännös

Ennenkuin havaitsemattomille vioille päästään tuottamaan testiä, on luonnollisesti tuotettava uusi scan-rakenteet sisältävä Cadat-tyyppinen kytkentälista, joka valmistellaan ja käännetään kuten aiemmin vastaavassa vaiheessa on tehty. Samoin on syytä ennen testingenerointia harkita tarkkaan tietämyskannan määritykset ja tehtävä myös sen käännös.

5.4.18 Testingenerointi havaitsemattomille vioille

Varsinainen testingenerointi käynnistetään vastaavalla komennolla kuin kohdassa 5.4.11.2, koska kysymyksessä on rajoitettu vikajoukko. Testingenerointia on syytä tässäkin tapauksessa tarkkailla mahdollisten virheiden havaitsemiseksi ajoissa. Tyypillisesti testingenerointi tässä vaiheessa havaisemattomille vioille on huomattavasti nopeampi kuin ensimmäinen testingenerointi.

5.4.19 Lisävektoreiden muunnos Silos-muotoon

Lisävektoreiden muunnos Silos2-muotoon tapahtuu kuten edellisessä vastaavassa vaiheessa. On kuitenkin huomattava, että viimeinen testingenerointi tehtiin T-solut fyysisesti sisältävälle kytkennälle, joten '-S'-valitsinta ei siis käytetä C2S-ohjelman ajossa. Tuotettua Silos-herätetiedostoa tarvitsee käsitellä ainoastaan, jos halutaan muuttaa testijakson pituutta.

5.4.20 Vikasimulointi Silos2:lla havaitsemattomille vioille lisävektoreilla

Vikasimulointi havaitsemattomille vioille tehdään Silos2:lla lisäherätteillä normaaliin tapaan. On muistettava määritellä Silos2:lle rajoitettu vikajoukko eli edellisen Silos2-vikasimuloinnin tuottama havaitsemattomien vikojen joukko. Tästä simuloinnista saatava havaitsemattomien vikojen joukko määrittelee lopullisen testinkattavuuden.

5.4.21 Testivektoritiedoston tuotto Silos2:lla

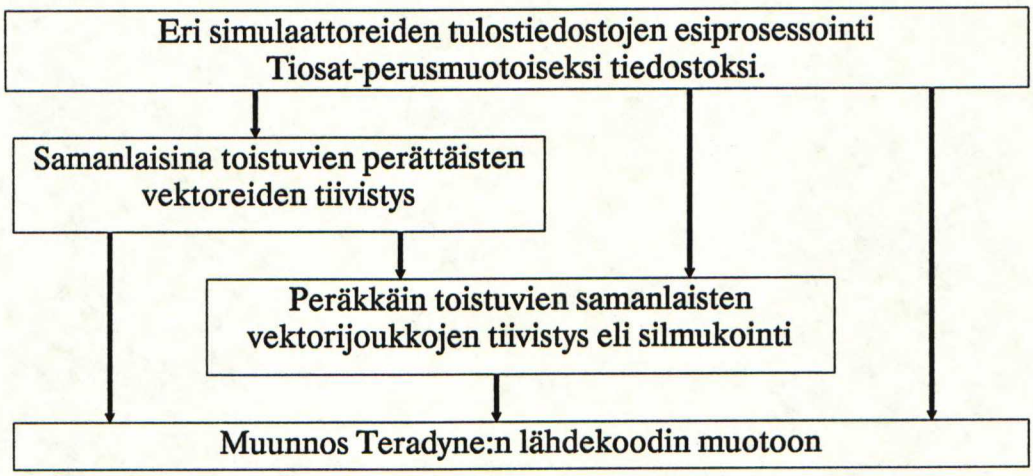
Tiosat-ohjelmistolle soveltuva testivektoritiedosto tuotetaan logiikkasimuloimalla kaikki käytetyt testiherätteet, mukaan lukien mahdolliset testivektoreina käytetyt toiminnalliset herätteet, ja yhdistämällä saadut tulostiedostot yhteen. Jos on käytetty T-soluja, ovat erivaiheissa saadut testiherätteet signaalien järjestyksen osalta eri muodossa, joten niiden yhteenliittäminen tekstieditorilla ei ole helppoa. Sen vuoksi on syytä tehdä logiikkasimuloinnit jokaiselle testiherätejoukolle erikseen, jonka jälkeen saadut tulostiedostot yhdistetään. Haluttaessa tulostiedostoja ei tarvitse yhdistää, vaan niille voidaan ajaa Tiosat-ajot jokaiselle erikseen.

5.4.22 Tiosat-ajo Teradyne-testivektoreiden tuottamiseksi.

Tiosat-ajolla tuotetaan Silos-tulostiedostosta Teradyne-testerin kääntäjälle sopiva lähdekielinen testikuviotiedosto.

5.5 TESTERILIITYNTÄ

Testeriliitynnän muodostaa ennen tätä diplomityötä tehty Tiosat-ohjelmisto, joka prosessoi viiden erilaisen logiikkasimulaattorin tulostiedostoista tietynlaisia tiivistysalgoritmeja ja muunnosalgoritmeja käyttäen Teradyne-testerin lähdekoodisen tiedoston. Tiosat-ohjelmiston eri vaiheet on esitetty kuvassa 5.7.



Kuva 5.7 Tiosat-ohjelmiston eri päävaiheet

5.6 KÄYTTÖESIMERKIT

5.6.1 Koekytkentä piilotetun scan-rakenteen havainnollistamiseksi

5.6.1.1 Kytkentä

Liitteessä 4 on esitettynä tämän esimerkin alkuperäinen kytkentäkaavio. Esimerkki muodostuu eräänlaisesta sarja-rinnakkaismuuntimesta, joka sisältää asynkronisia jakajia, siirtorekisteriketjun, rinnakkaisrekisterin ja kaksisuuntaisen väylän. Liitteessä 5 on esitettynä kytkentä, johon on lisätty erityisrakenteet piilotetun scan-ketjun toteuttamiseksi eli 12 porttia, yksi ylimääräinen tulo ja yksi multiplekseri. Liitteessä 6 on esitettynä lopullinen kytkentä, johon on lisätty varsinaiset T-solut ja niiden ohjaussolu.

5.6.1.2 Generointiprosessi

Generointiprosessi aloitettiin tekemällä testattavuusanalyysi koko kytkennälle. Tuloksista todettiin, että asettamalla T-solut solun U01103 lähtöön ja solun U0160 Q-lähtöön, saadaan huonommin ohjattavan solmun ohjattavuusluku pienennettyä arvosta 42516 arvoon 18804.

Tietämyskantaan tehtiin seuraavat määrittelyt:

```
#machine
    tbit = test;
#illegal
    tbit:0;
#zero
    1:test;
#path
    u0199(u1.q),u0194(u1.y0);
    u016(u1.qn),u016(u1.q);
    u015(u1.qn),u015(u1.q);
```

Määrittelyissä pakotetaan Test-signaali ensimmäisestä vektorista alkaen tilaan 1, ja lisäksi opastetaan Intelligeniä paremman polun valitsemisessa.

Seuraavaksi suoritettiin T-solumääritysten käännös .TLF-tiedoston sisältäessä seuraavan informaation T-solujen paikoiksi:

```
u01103(u1.y0)
u0160(u1.q)
```

Seuraavaksi lisättiin kytkentälistaan Parts-kenttään 'Testio Testio $', jonka jälkeen kytkentälista oli liitteen 7 mukainen. Suoritettiin kytkentälistan käännös uudelleen ja käynnistettiin testingenerointi Intelligenin valikosta. Tässä tapauksessa ei käytetty karsittua vikajoukkoa, koska kytkennälle ei ollut toiminnallisia simulointivektoreita lainkaan.

Testingenerointiin kului CPU-aikaa 162s ja testin kattavuudeksi saatiin 70.8%. Generoinnin kulkua selvittää liite 8.

Suoritettiin Silos-muotoisten vektoreiden tuottaminen C2S-ohjelmalla .HOW-tiedoston ollessa:

xin5,xout4,xin3,xin4;.

Silos-muotoisessa herätetiedostossa scan-ohjaukseen tarkoitetut signaalit esiintyvät .HOW-tiedostossa esitetyillä nimillä, ja lisäksi siinä on esitettynä todellisten tulojen in5, out4, in3 ja in4:n tilat. Editoimalla scan-ohjauksen nimet todellisiksi nimiksi ja todelliset nimet dummy1:ksi jne, saatiin scan-ohjaukset ohjatuiksi oikeisiin tuloihin. Osa herätetiedostoa on esitetty liitteessä 9.

Seuraavaksi lisättiin T-solut kytkentäkaavioon ja tuotettiin uudet Silos- ja Intelligen-muotoiset kytkentälistat.

Kytkentä vikasimuloitiin Silos2:lla karsimattomalla vikajoukolla ja kattavuudeksi saatiin 77% simulointiajan ollessa n. 4 min. Ero kattavuudessa Intelligenin tulokseen nähden johtuu mm. erilaisesta redundanttisten vikojen käsittelytavasta. Vikasimuloinnin alussa annettiin määritys '.SYMBOL R1 = Z', joka määrittelee herätetiedostossa olevan 'Z':n resistiiviseksi '1':ksi korvaamaan Silos-mallituksesta puuttuvan IOL4:n ylösvetovastuksen.

Vikasimuloinnin tuloksena saatiin liitteen 10 mukainen havaitsemattomien vikojen lista, josta käsin editoitiin liiteen 11 mukainen .FSL-tiedosto seuraavan Intelligen-ajon vikajoukoksi. Vikajoukosta on karsittu pois IOC4- ja IOL4-nimisten solujen CP-nimisen välisignaalin s-a-1-vika, joka ei paljastu johtuen ylösvedosta IOL4-solussa. Vika paljastuu kuitenkin viimeistään IOL4-lähtöjen virranantokykytestissä.

Seuraava Intelligen-ajo rajoitetulla vikajoukolla vei 9s CPU-aikaa, ja siitä saatiin kattavuudeksi rajoitetun vikajoukon osalta 100%. Generoinnin kulkua selvittää liite 12.

C2S-ohjelmalla tuotettiin lisäherätetiedosto.

Suoritettiin Silos2-simulointi edellisestä Silos2-vikasimuloinnista saadulle karsitulle vikajoukolle lisävektoreilla ja saatiin kattavuudeksi 100%.

Seuraavaksi suoritettiin logiikkasimulointi sekä ensimmäisen generointikierroksen tuottamilla herätteillä että lisäherätteillä ja liitettiin kummastakin simuloinnista saadut Tiosat-ohjelmistolle tarkoitetut tulostiedostot yhteen. Liitteessä 13 on osa tulostiedostoa.

Lopuksi suoritettiin Tiosat-ajo ja tuloksena saatiin 2607 vektoria sisältävä lähdekielinen testikuviotiedosto testerille. Liitteessä 14 on osa testikuviotiedostoa.

5.6.2 Koeajoissa käytetyn Pilot-kytkennän modifioitu osa

5.6.2.1 Kytkentä

Tässä esimerkkiajossa käytetään kytkentänä Pilot-kytkennän vastaanotinosaa, joka muodostuu FSK-ilmaisimesta, digitaalisesta vaihelukosta, kehyssynkronoinnin suorittavasta logiikasta, rinnakkaisrekisteristä, asynkronisesta laskurista kombinaatiologiikkoineen, FSK-modulaation tunnistimesta ja kaksisuuntaisesta primäärivaylästä. Kytkentäkaaviot on esitetty liitteessä 15.

Kytkentä on alunperin suunniteltu manuaalista testingenerointia ajatellen, ja siten se sisälsi kohtuullisen paljon erilaisia testattavuutta parantavia testirakenteita. Tätä esimerkkiä varten kytkennästä poistettiin kaikki testirakenteet ja ryhdyttiin parantamaan testattavuutta ainoastaan T-solujen ja Idisp-testattavuusanalyysin avulla. T-soluja käytettiin yhteensä 11 kappaletta. Lisäksi tarvittiin kolme ylimääräistä tuloa ja yksi ylimääräinen lähtö, koska tässä tapauksessa ei käytetty piilotettua scan-rakennetta. Piirin pinta-ala kasvoi noin 6.5% verrattuna alkuperäiseen kytkentään.

Kytkennälle tehtiin ohjeiston mukaisesti testattavuusanalyysit jokaiselle lohkolle erikseen. Testattavuusanalyysin jälkeen suoritettiin jokaiselle lohkolle testingenerointi, jotta mahdolliset ongelmapaikat olisivat löytyneet.
Testattavuusanalyysien ja testingenerointien tulokset on esitetty taulukossa 5.1

Lohko	OE	OJ	n	Kattavuus
M11	45	-	0	100%
M13	17982	771	3	97.2%
M14	280	-	0	100%
M15	5	-	0	100%
M17	2888	2888	2 *)	100%
M18	13315	1319	2	100%
M20	9	-	0	94.0%

OE = huonoin ohjattavuus ennen modifiointia
OJ = huonoin ohjattavuus modifioinnin jälkeen
n = T-solujen määrä
*) Tähän lohkoon asetetut kaksi T-solua eivät paranna huonoimmin ohjattavien solmujen ohjattavuutta. Toinen katkaisee kahdeksanbittisen laskurin ja toinen katkaisee asynkronisen nollaussignaalin.

Taulukko 5.1

Liitteessä 15 esitetyissä kytkentäkaavioissa on merkittyinä paikat, joihin T-solut on tarkoitus asettaa. Perustelut T-solujen paikkojen valinnoille ovat:

M13:

T-solut kokosummaimen u0122 ja puolisummaimen u0123 tuloissa auttavat Intelligeniä huomattavasti summaimien lähtöjen ohjaamisessa. Käytännössä testingenerointi ilman näitä T-soluja ei onnistuisi.

T-solu u0113:n lähdössä ainoastaan lisää solmun ohjattavuutta ja havaittavuutta.

M17:

T-solu u014:n lähdössä katkaisee kahdeksanbittisen laskurin, jonka ohjaaminen olisi muutoin ehkä liian paljon aikaakuluttavaa Intelligenille.

T-solu u0139:n lähdössä katkaisee aynkronisen nollaustoiminnon, joka muutoin estäisi tiettyjen vikojen testingeneroinnin.

M18:

Tässä lohkossa käytetyt T-solut ainoastaan parantavat solmujen ohjattavuutta ja havaittavuutta.

Koska testattavuuden tarkastelussa saadut tulokset ovat riittävän hyviä, siirryttiin koko piirin testattavuuden tarkasteluun. Tuloksena päätettiin asettaa kytkentään vielä 4 T-solua lisää kytkentäkaavioissa esitettyihin paikkoihin, jolloin huonoin ohjattavuus parani 3129551:stä 10475:een. Nyt määritellyt 4 uutta T-solua ainoastaan lisäävät kytkennän ohjattavuutta ja havaittavuutta.

Testingenerointi käynnistettiin normaaliin tapaan ilman tietämyskantamäärittelyjä. Testingenerointi vei CPU-aikaa noin 20 minuuttia ja kattavuudeksi tuli 95.9%. Generoinnin kulkua selvittää liite 16.

Generoinnin jälkeen tuotettiin C2S-ohjelmalla Silos-muotoiset testiherätteet, lisättiin T-solut kytkentäkaavioihin ja tuotettiin uudet kytkentälistat. Lopulliset kytkentäkaaviot on esitetty liitteessä 17.

Silos2:lla suoritetusta vikasimuloinnista saatiin havaitsemattomien vikojen joukko, joka muunnettiin Cadat-muotoon seuraavaa Intelligen-ajoa varten. Alkuperäinen vikajoukko on esitettynä liitteessä 18.

Seuraava Intelligen-ajo vei aikaa noin 10 min ja tuotti testin neljälle aiemmin havaitsematta jääneelle vialle. Saatu testiheräte muunnettiin Silos2-muotoon, ja sillä suoritettiin vikasimulointi edellisessä Silos2-simuloinnissa havaitsematta jääneille vioille. Tuloksena saatiin 24 havaitsematonta vikaa, joka antaa testin lopulliseksi kattavuudeksi 95.7%. Generoinnin kulkua selvittää liite 19, ja havaitsemattomat viat on lueteltuna liitteessä 20.

6 ATPG-OHJELMAN ASETTAMAT RAJOITUKSET LOGIIKKASUUNNITTELULLE

6.1 KOMBINAATIOLOGIIKKA

6.1.1 Takaisinkytkennät

Automaattinen testingenerointi ei aiheuta varsinaisia rajoituksia kombinaatiologiikan suunnittelulle. On kuitenkin muistettava, että takaisinkytkentä kombinaatiologiikassa saattaa muuttaa kytkennän luonteen sekvenssilogiikaksi. Tällaisia rakenteita ovat esimerkiksi erillisillä porteilla toteutetut RS-kiikut, joiden käsittely on Intelligenille erityisen hankalaa. Perusohjeena voidaan pitää sitä, että kombinaatiologiikalla ei saa toteuttaa minkäänlaisia sekvenssielimiä. Jos haluttua sekvenssielintä ei voida toteuttaa kuin kombinaatiologiikan malleilla, on sille tehtävä oma Cadat-malli, jolloin Intelligenillä on paremmat mahdollisuudet sen toiminnan ymmärtämiseen.

On havaittu, että kannattaa aina, kun vain on mahdollista, järjestää takaisinkytkentäsilmukkaan jonkinlainen katkaisumahdollisuus. Silloin kombinaatiologiikka on puhdasta totuustaululogiikkaa ja testingenerointi on helppoa.

6.2 SEKVENSSILOGIIKKA

6.2.1 Synkroninen ja asynkroninen logiikka

Erilaisissa kokeiluissa on todettu, että Intelligen pystyy tuottamaan testin helpommin synkroniselle logiikalle kuin asynkroniselle logiikalle. Intelligen kuitenkin pystyy, kuten esimerkeistä ilmenee, tuottamaan testin asynkroniselle logiikalle. Kuitenkin prosessointiajat ovat synkroniselle logiikalle lyhyempiä.

6.2.2 Asynkroniset nollauskytkennät

Erityisen vakavan ongelman Intelligenille muodostaa esimerkiksi rakenne, jossa asynkronisen laskurin tiloista dekoodattava signaali nollaa laskurin. Tällaisessa tapauksessa on laskurilla olemassa tila, joka ei koskaan ole pysyvä, vaan aiheuttaa nollautumisen automaattisesti. Näin ollen Intelligen saattaa tuottaa herätteet mainitun tilan saavuttamiseksi, mutta todellisuudessa tila ei koskaan esiinny, jolloin testi epäonnistuu.

6.2.3 Suuret laskurit

Suuret laskurikytkennät eivät ole este Intelligenin testingeneroinnille, mutta niiden vaatimien pitkien herätteiden tuottaminen vie yleensä huomattavasti aikaa. Esimerkiksi 8-bittinen laskuri vaatii vähintään 512 testivektoria testaukseen, mutta kaksi erillistä 4-bitin laskuria vain 32 vektoria kumpikin. Yleisohjeena voidaan pitää laskurin bittimäärän maksimina 4:ää, jolloin pidemmät laskurit on jaettava osiin joko erityisellä testirakenteella tai T-solulla.

6.2.4 Totuustaulut

Totuustaulujen käytössä on havaittu olevan joissain tapauksissa vaikutus testingeneroinnin nopeuteen. Kysymyksessä on tapauskohtainen asia, jossa tietämyskannan avulla kannattaa tutkia, missä järjestyksessä Intelligen käyttää kyseessä olevan primitiivin totuustaulun rivejä.

7 YHTEENVETO

Työssä täydennettiin Micronas Oy:n suunnittelujärjestelmää digitaalikytkentöjen testivektoreiden automaattisella generointimenetelmällä. Työkalu sijoittuu luontevasti muiden työkalujen joukkoon ja sen on tarkoitus olla jokaisen digitaalisuunnittelijan käytössä.

Työ painottui tasaisesti alkuosan algoritmien selvityksen ja loppuosan käyttöönoton ja menetelmäkehityksen kesken.

Tähän mennessä saatujen kokemusten pohjalta voidaan todeta, että järjestelmä tulee parantamaan Micronas Oy:n testingenerointikapasiteettia ja samalla testien kattavuutta huomattavasti. Intelligeniä käyttäen testingenerointi on useita kertaluokkia manuaalista testingenerointia nopeampi. Erityisesti T-solujen käyttö tuo lisää tehokkuutta.

Seuraavassa Intelligenin versiossa on mahdollista käyttää myös ylemmän hierarkiatason mallituksessa totuustauluja, jolloin testingenerointi nopeutuu entisestäänkin. Lisäksi on todennäköistä, että tietämyskannan ominaisuuksia parannetaan, jolloin on mahdollista määritellä piirin sisäiselle solmulle yksi ja vain yksi tila (nykyisellään se ei ole mahdollista). Silloin piilotetun scan-rakenteen ohjaussignaali voidaan tuottaa rekisterillä, joten ei tarvita yhtään ylimääräistä tuloa.

8 LÄHTEET

1. Fujiwara, H. Logic testing and design for testability 2.p. Massachusetts, USA 1985 MIT 279 s.

2. Bennets, R.G. Design of testable logic circuits 2.p. Lontoo 1983 Addison-Wesley Publishing Company 164 s.

3. Kirkland, T. & Mercer, M. Algorithms for automatic test pattern generation IEEE Design & Test of Computers 5 (1988) 3 s. 43 - 55

4. Marlett, R. Automated test generation for integrated circuits VLSI System Design 7 (1986) 7 s. 68 - 73

5. Zycad Corporation Nextgen II users manual Minnesota, USA 1988 Zycad Corporation 162 s.

6. HHB Systems Intelligen 1.0 Documentation package New Jersey, USA 1988 HHB Systems 22 s.

7. HHB Europe Theseus-esite

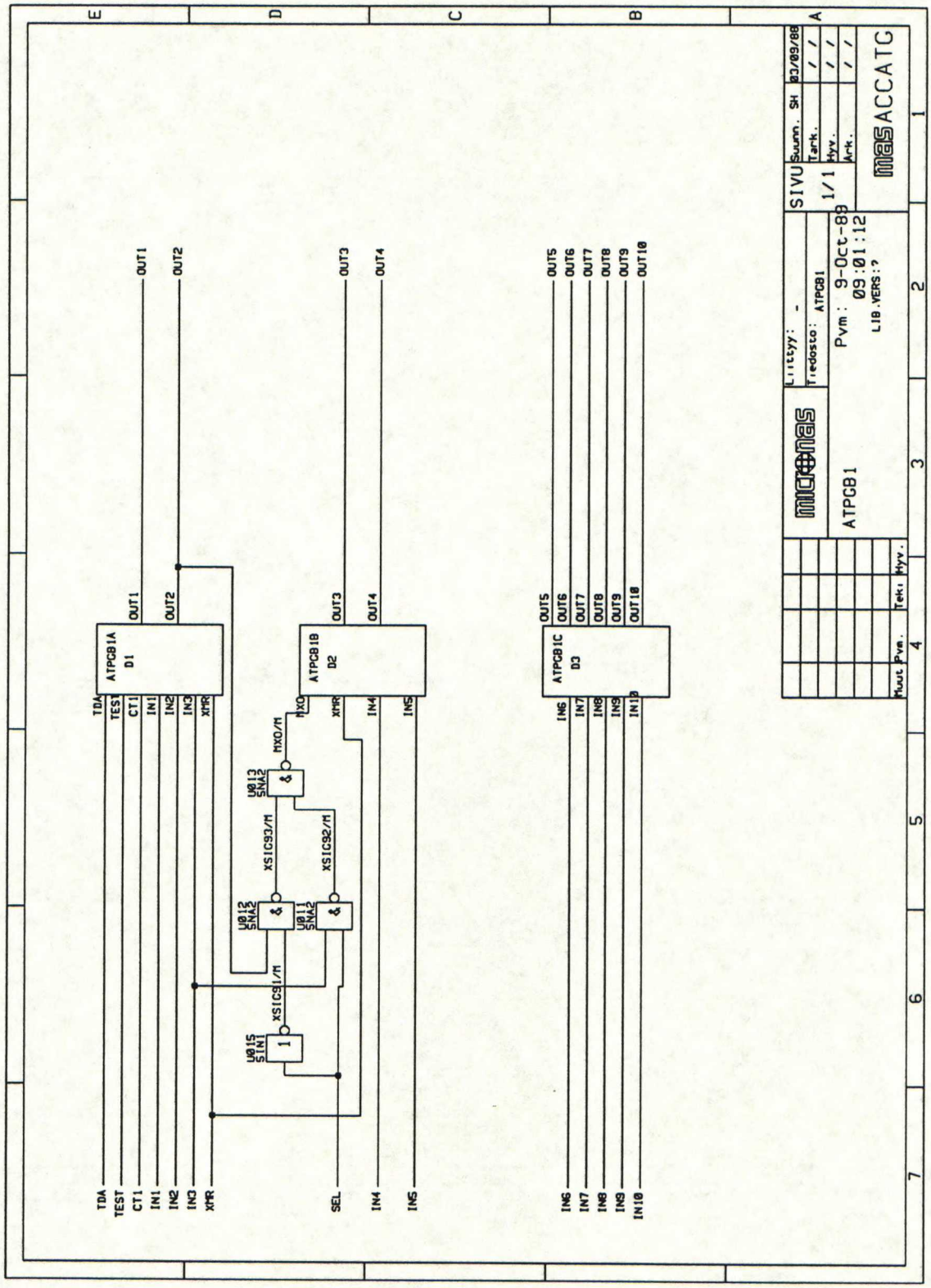
8. Zycad Corporation Nextgen-esite

9. Bennets, R.G. Testing digital electronics The Center for Professional Advancement s. E1-E30

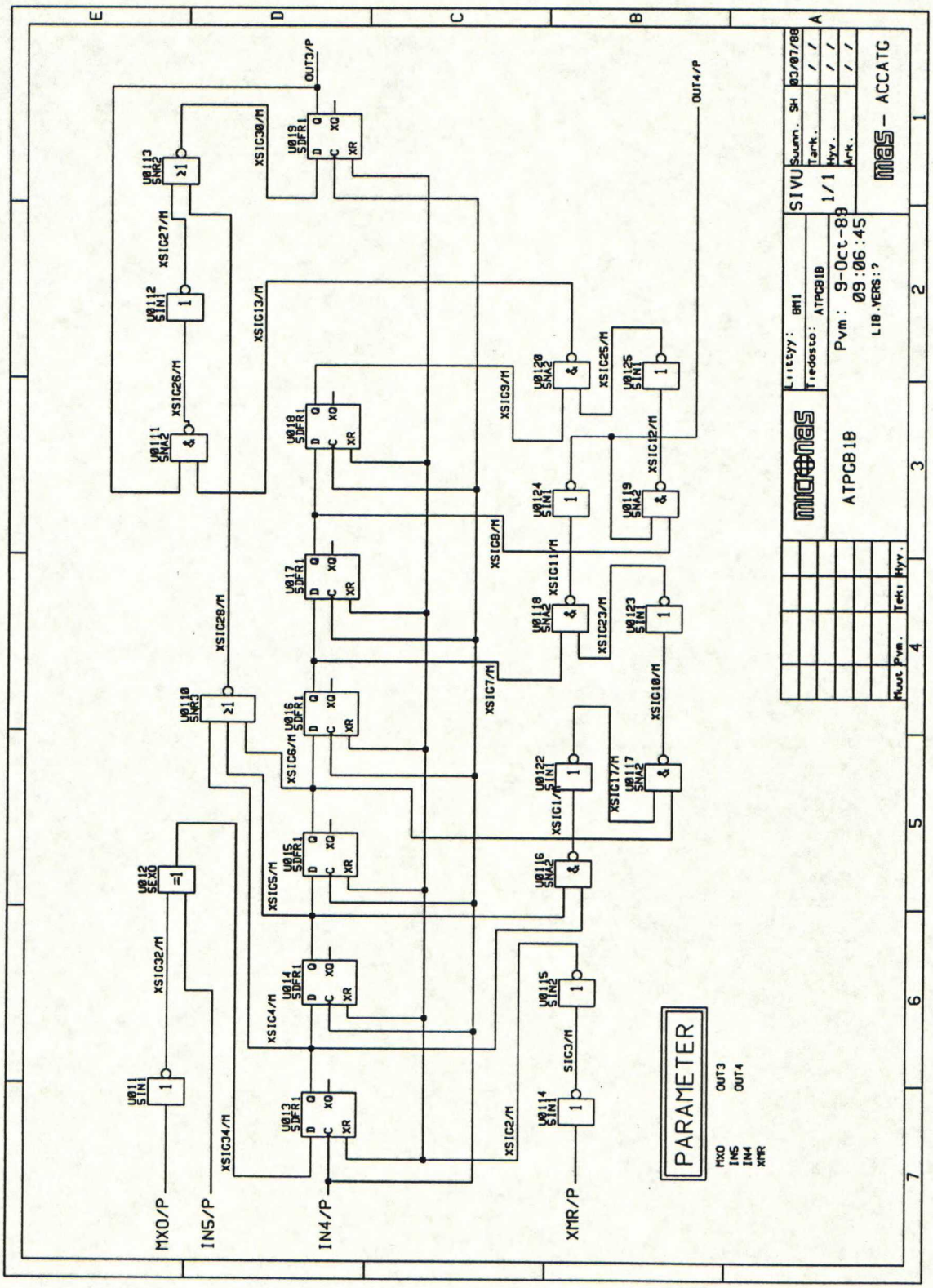
10. Nokia tutkimuskeskus Atte user's manual version 3.0 Espoo 1989 Nokia tutkimuskeskus

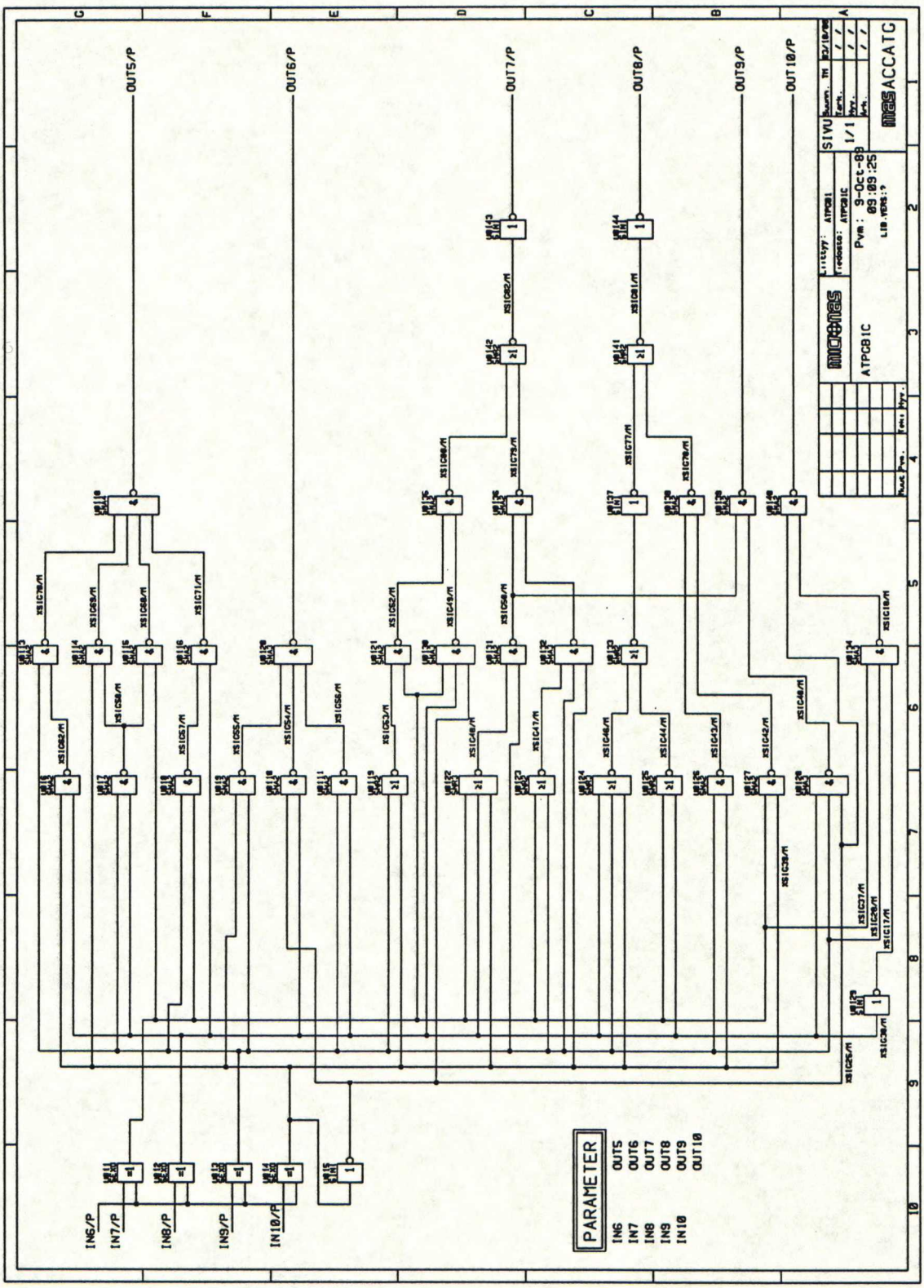
11. Valtion teknillinen tutkimuskeskus Automaattinen testivektoreiden generointi

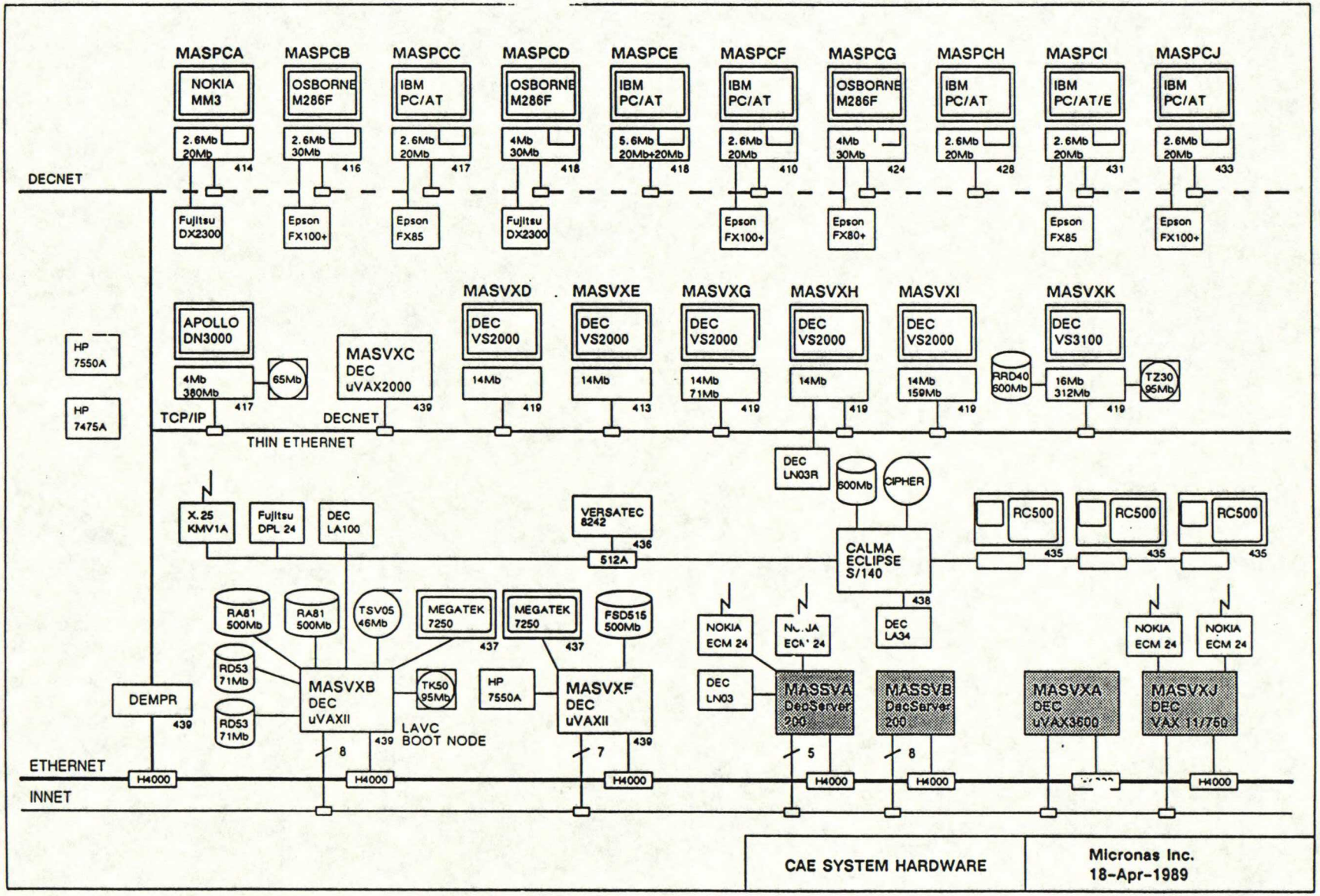
12. Valtion teknillinen tutkimuskeskus Kaupalliset testivektorigeneraattorit

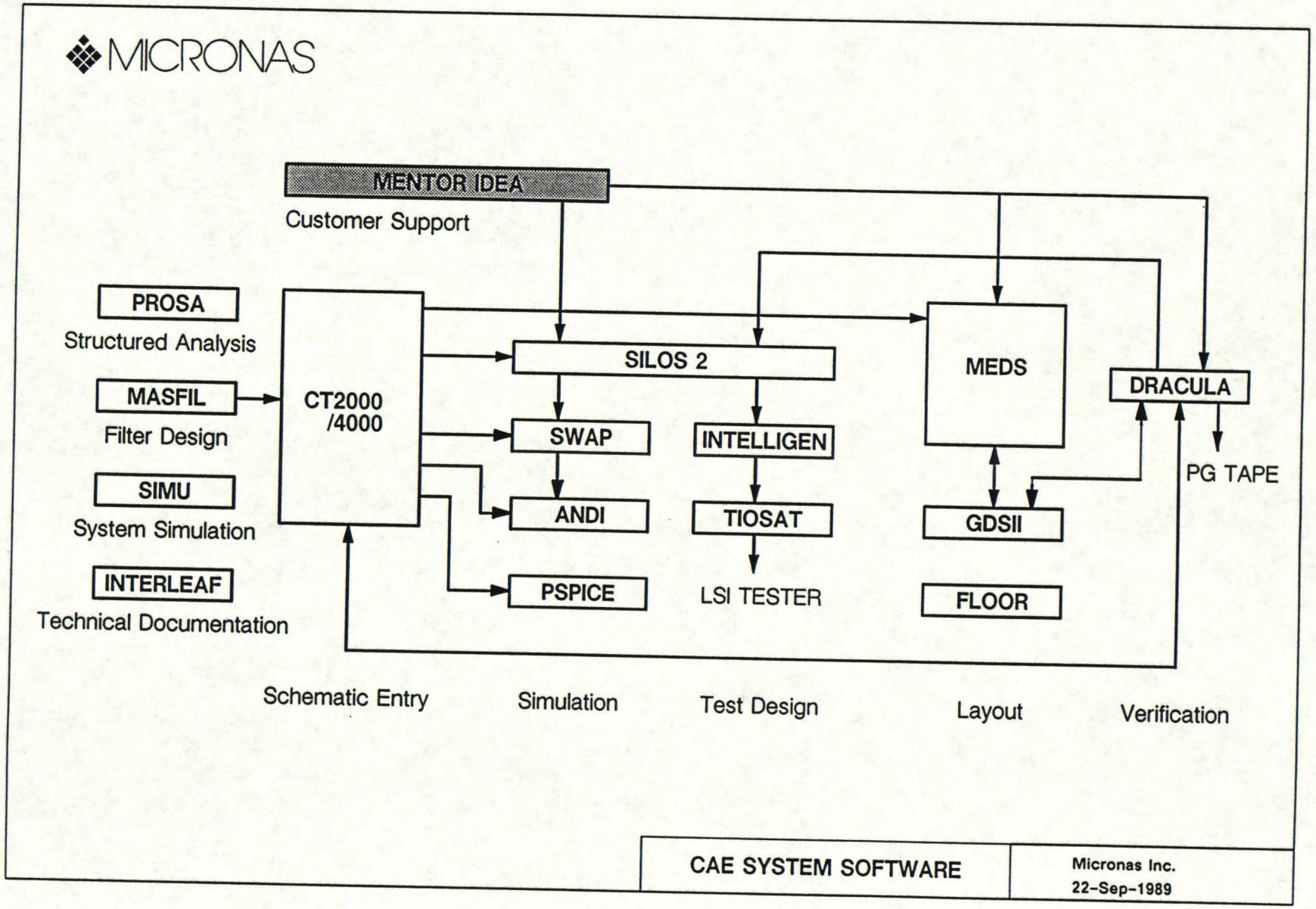

ATPCB1A
D1
ATPCB1B
D2
ATPCB1C
D3
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
OUT9
OUT10
TDA
TEST
CT1
IN1
IN2
IN3
XMR
MXO
IN4
IN5
IN6
IN7
IN8
IN9
IN10
SEL
XSIC93/M
XSIC92/M
XSIC91/M
U813 SNA2
U812 SNA2
U811 SNA2
U815 SIN1
ATPCB1
Piirtäjä: -
Tiedosto: ATPCB1
Pvm: 9-Oct-89
09:01:12
LIB.VERS:?
SIVU 1/1
Suunn. SH 03/09/88
Tark. / /
Hyv. / /
Ark. / /
Muut Pvm. Tekl Hyv.
masACCATG

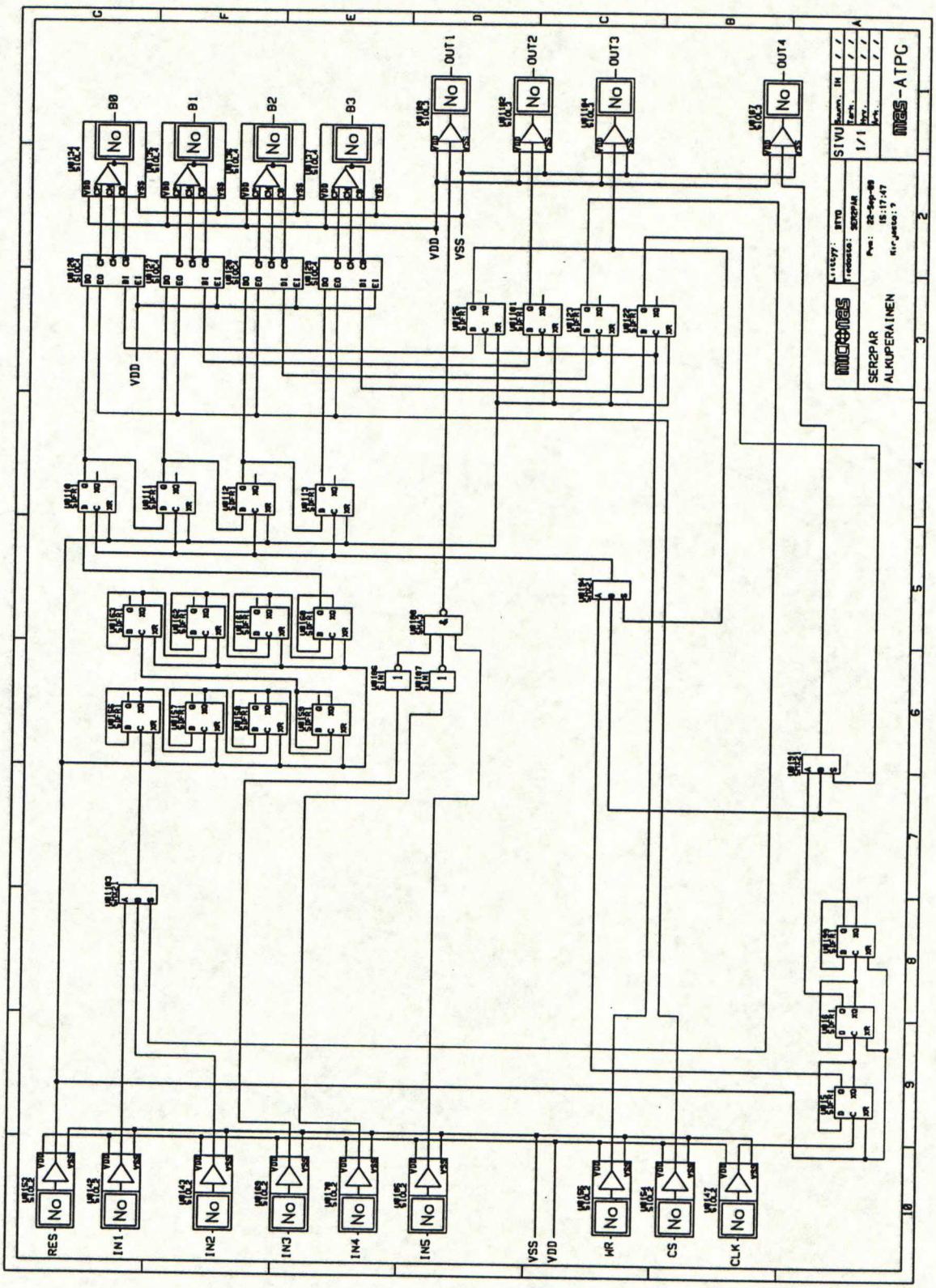
OUT2/P
OUT1/P
TDA/P
TEST/P
CT1/P
IN3/P
IN2/P
IN1/P
XTR/P
PARAMETER
TDA
TEST
CT1
IN3
IN2
IN1
XTR
OUT2
OUT1
ATPCB1A
S1VU
9-Oct-98
09:03:14
ACCATG

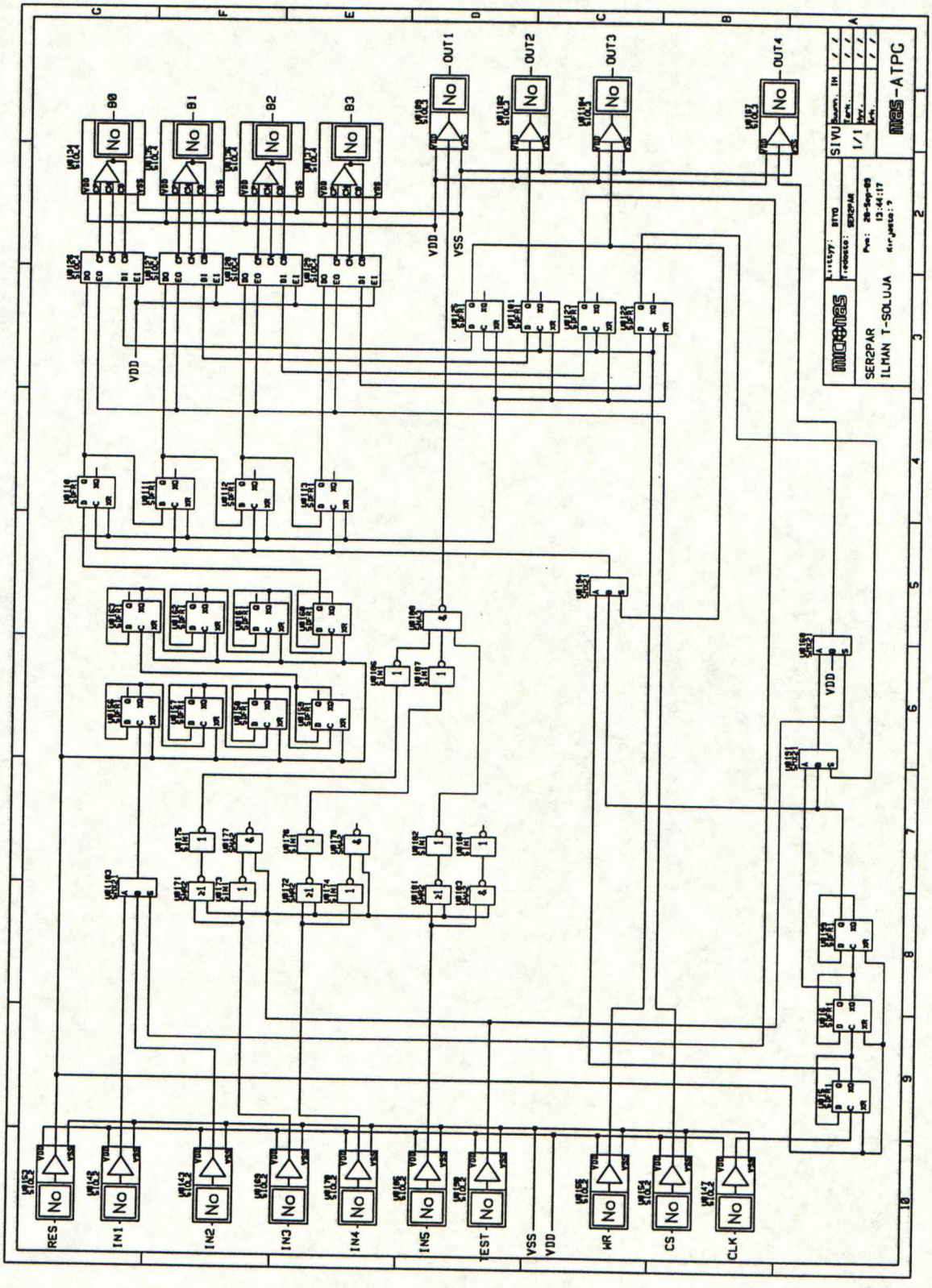

MXO/P
INS/P
IN4/P
XMR/P
OUT3/P
OUT4/P
PARAMETER
MXO OUT3
INS OUT4
IN4
XMR
ATPCB1B
Pvm: 9-Oct-88
09:06:45
LIB.VERS:?
SIVU 1/1
MAS - ACCATG

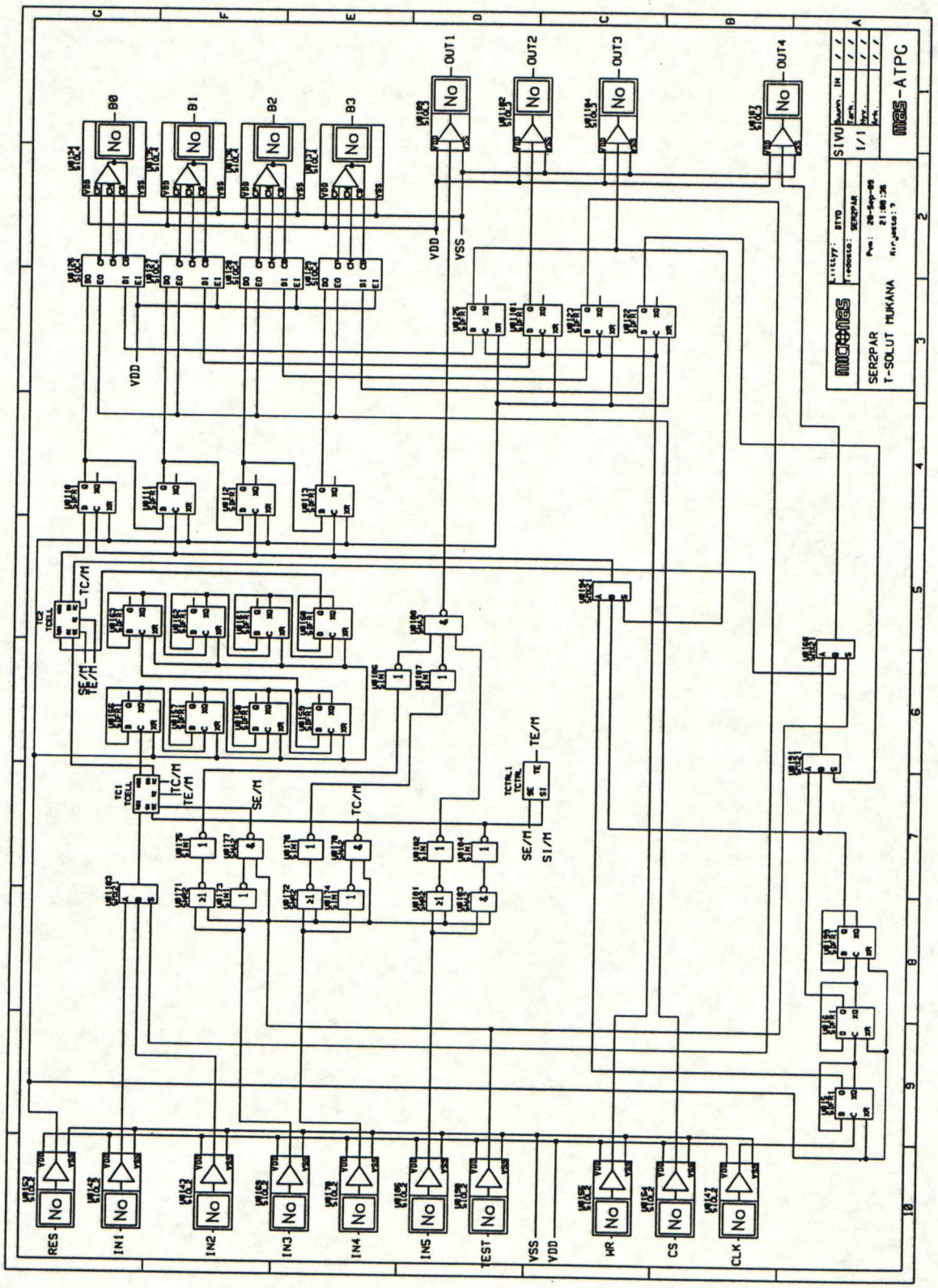

OUT5/P
OUT6/P
OUT7/P
OUT8/P
OUT9/P
OUT10/P
IN6/P
IN7/P
IN8/P
IN9/P
IN10/P
PARAMETER
IN6 OUT5
IN7 OUT6
IN8 OUT7
IN9 OUT8
IN10 OUT9
OUT10
ATPCB1C
SIVU 1/1
ACCATG


MASPCA
NOKIA MM3
2.6Mb 20Mb
414
MASPCB
OSBORNE M286F
2.6Mb 30Mb
416
MASPCC
IBM PC/AT
2.6Mb 20Mb
417
MASPCD
OSBORNE M286F
4Mb 30Mb
418
MASPCE
IBM PC/AT
5.6Mb 20Mb+20Mb
418
MASPCF
IBM PC/AT
2.6Mb 20Mb
410
MASPCG
OSBORNE M286F
4Mb 30Mb
424
MASPCH
IBM PC/AT
2.6Mb 20Mb
428
MASPCI
IBM PC/AT/E
2.6Mb 20Mb
431
MASPCJ
IBM PC/AT
2.6Mb 20Mb
433
DECNET
Fujitsu DX2300
Epson FX100+
Epson FX85
Fujitsu DX2300
Epson FX100+
Epson FX80+
Epson FX85
Epson FX100+
MASVXD
MASVXE
MASVXG
MASVXH
MASVXI
MASVXK
APOLLO DN3000
4Mb 380Mb
417
65Mb
MASVXC DEC uVAX2000
439
DEC VS2000
14Mb
419
DEC VS2000
14Mb
413
DEC VS2000
14Mb 71Mb
419
DEC VS2000
14Mb
419
DEC VS2000
14Mb 159Mb
419
RRD40 600Mb
DEC VS3100
16Mb 312Mb
419
TZ30 95Mb
HP 7550A
HP 7475A
TCP/IP
DECNET
THIN ETHERNET
DEC LN03R
600Mb
CIPHER
X.25 KMV1A
Fujitsu DPL 24
DEC LA100
VERSATEC 8242
436
512A
CALMA ECLIPSE S/140
438
RC500
435
RC500
435
RC500
435
RA81 500Mb
RA81 500Mb
TSV05 46Mb
MEGATEK 7250
437
MEGATEK 7250
437
FSD515 500Mb
RD53 71Mb
RD53 71Mb
MASVXB DEC uVAXII
TK50 95Mb
439
LAVC BOOT NODE
HP 7550A
MASVXF DEC uVAXII
439
NOKIA ECM 24
NOKIA ECM 24
DEC LA34
NOKIA ECM 24
NOKIA ECM 24
DEC LN03
MASSVA DecServer 200
MASSVB DecServer 200
MASVXA DEC uVAX3600
MASVXJ DEC VAX 11/750
DEMPR
439
8
7
5
8
ETHERNET
INNET
H4000
CAE SYSTEM HARDWARE
Micronas Inc.
18-Apr-1989


MICRONAS
MENTOR IDEA
Customer Support
PROSA
Structured Analysis
MASFIL
Filter Design
SIMU
System Simulation
INTERLEAF
Technical Documentation
CT2000
/4000
SILOS 2
SWAP
ANDI
PSPICE
INTELLIGEN
TIOSAT
LSI TESTER
MEDS
GDSII
FLOOR
DRACULA
PG TAPE
Schematic Entry
Simulation
Test Design
Layout
Verification
CAE SYSTEM SOFTWARE
Micronas Inc.
22-Sep-1989


B0
B1
B2
B3
OUT1
OUT2
OUT3
OUT4
VDD
VSS
RES
IN1
IN2
IN3
IN4
IN5
WR
CS
CLK
SERZPAR
ALKUPERAINEN
SIVU
1/1
mas-ATPC


SER2PAR
ILHAN T-SOLUJA
SIVU
1/1
mas-ATPG
OUT1
OUT2
OUT3
OUT4
B0
B1
B2
B3
VDD
VSS
RES
IN1
IN2
IN3
IN4
IN5
TEST
HR
CS
CLK


OUT1
OUT2
OUT3
OUT4
B0
B1
B2
B3
VDD
VSS
TC/M
SE/M
TE/M
SI/M
RES
IN1
IN2
IN3
IN4
IN5
TEST
WR
CS
CLK
SIVU
SER2PAR
T-SOLUT MUKANA
mas-ATPG

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  ser2par.dat  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                    N e t w o r k   T r a n s l a t i o n
$
$  S I L O S  I I    ver 1.005        Wed Sep 20 13:50:43 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

HEADER $

    CIRCUIT NAME $
        File name = "ser2par.ckt" $

    DEFAULT DELAY $
        lns, lns $

    TECHNOLOGY $
        apa $

    COMMENTS $
        Time scale converted at 1 Silos unit = 1 ns $

PARTS $
    IC PARTS $
        TESTIO TESTIO $
        u0199 dfr1 $
        u0194 mx21 $
        u0191 mx21 $
        u0190 iol2 $
        u0189 iol3 $
        u0188 na3 $
        u0187 in1 $
        u0186 in1 $
        u0185 iol2 $
        u0184 in1 $
        u0183 na2 $
        u0182 in1 $
        u0181 nr2 $
        u0178 na2 $
        u0177 na2 $
        u0176 in1 $
        u0175 in1 $
        u0174 in1 $
        u0173 in1 $
        u0172 nr2 $
        u0171 nr2 $
        u0170 iol2 $
        u0169 iol2 $
        u0168 mx21 $
        u0167 iol3 $
        u0163 dfr1 $
        u0162 dfr1 $
        u0161 dfr1 $
        u0160 dfr1 $
        u016 dfr1 $
        u0159 dfr1 $
        u0158 dfr1 $
        u0157 dfr1 $
        u0156 dfr1 $
        u0155 iol2 $
        u0154 iol2 $
        u0153 iol2 $
        u015 dfr1 $
        u0147 iol2 $
        u0143 iol2 $
        u0142 iol2 $
        u0137 iol4 $
        u0136 iol4 $
        u0135 iol4 $
```

```
        u0134 iol4 $
        u0129 ioc4 $
        u0128 ioc4 $
        u0127 ioc4 $
        u0126 ioc4 $
        u0125 dfr1 $
        u0123 dfr1 $
        u0122 dfr1 $
        u0113 dfr1 $
        u0112 dfr1 $
        u0111 dfr1 $
        u01104 iol3 $
        u01103 mx21 $
        u01102 iol3 $
        u01101 dfr1 $
        u0110 dfr1 $

CONNECTIONS $
$   u0159%d
        u0163.2
+       u0159.1
+       u0159.5
+       "u0159%d"  $  3
$   .unk
        $ no connections
$   .vcc
        $ no connections
$   u0181%q
        u0182.1
+       u0181.3
+       "u0181%q"  $  2
$   .vss
        $ no connections
$   u0182%q
        u0188.3
+       u0182.2
+       "u0182%q"  $  2
$   u0183%q
        u0184.1
+       u0183.3
+       "u0183%q"  $  2
$   res
        u0153.1
+       res  $  1
$   u0128%cb
        u0136.3
+       u0128.7
+       "u0128%cb"  $  2
$   u0128%cn
        u0136.2
+       u0128.6
+       "u0128%cn"  $  2
$   u0128%cp
        u0136.1
+       u0128.5
+       "u0128%cp"  $  2
$   u0184%q
        u0184.2
+       "u0184%q"  $  1
$   u0110%q
        u0126.1
+       u0111.1
+       u0110.4
+       "u0110%q"  $  3
$   u0110%c
        u0194.4
+       u0113.2
+       u0112.2
```

```
+       u0111.2
+       u0110.2
+       "u0110%c"  $  5
$   u0110%d
        u0160.4
+       u0110.1
+       "u0110%d"  $  2
$   u0186%q
        u0188.1
+       u0186.2
+       "u0186%q"  $  2
$   b0
        u0134.4
+       b0I
+       b0O  $  1
$   b1
        u0135.4
+       b1I
+       b1O  $  1
$   b2
        u0136.4
+       b2I
+       b2O  $  1
$   b3
        u0137.4
+       b3I
+       b3O  $  1
$   u0111%q
        u0127.1
+       u0112.1
+       u0111.4
+       "u0111%q"  $  3
$   cs
        u0154.1
+       cs  $  1
$   u0187%q
        u0188.2
+       u0187.2
+       "u0187%q"  $  2
$   u0160%c
        u0161.1
+       u0161.5
+       u0160.2
+       "u0160%c"  $  3
$   u0160%d
        u0160.1
+       u0160.5
+       "u0160%d"  $  2
$   vdd
        u0168.2
+       u0129.4
+       u0128.4
+       u0127.4
+       u0126.4
+       vdd  $  5
$   u0112%q
        u0128.1
+       u0113.1
+       u0112.4
+       "u0112%q"  $  3
$   u0167%in1
        u0168.4
+       u0167.1
+       "u0167%in1"  $  2
$   wr
        u0155.1
+       wr  $  1
$   u0188%q
```

```
        u0189.1
+       u0188.4
+       "u0188%q"  $  2
$   u0161%c
        u0162.1
+       u0162.5
+       u0161.2
+       "u0161%c"  $  3
$   u0113%q
        u0129.1
+       u0113.4
+       "u0113%q"  $  2
$   u0162%c
        u0163.1
+       u0163.5
+       u0162.2
+       "u0162%c"  $  3
$   u0126%cb
        u0134.3
+       u0126.7
+       "u0126%cb"  $  2
$   u0126%cn
        u0134.2
+       u0126.6
+       "u0126%cn"  $  2
$   u0126%cp
        u0134.1
+       u0126.5
+       "u0126%cp"  $  2
$   test
        u0190.1
+       test  $  1
$   u0126%eo
        u0154.2
+       u0129.2
+       u0128.2
+       u0127.2
+       u0126.2
+       "u0126%eo"  $  5
$   u0110%xr
        u0199.3
+       u0163.3
+       u0162.3
+       u0161.3
+       u0160.3
+       u016.3
+       u0159.3
+       u0158.3
+       u0157.3
+       u0156.3
+       u0153.2
+       u015.3
+       u0125.3
+       u0123.3
+       u0122.3
+       u0113.3
+       u0112.3
+       u0111.3
+       u01101.3
+       u0110.3
+       "u0110%xr"  $  20
$   u0168%s
        u0190.2
+       u0183.2
+       u0181.2
+       u0178.2
+       u0177.2
+       u0172.2
```

```
+       u0171.2
+       u0168.3
+       "u0168%s"  $  8
$   u0168%a
        u0191.4
+       u0168.1
+       "u0168%a"  $  2
$   u0169%q
        u0173.1
+       u0171.1
+       u0169.2
+       "u0169%q"  $  3
$   u0191%b
        u0199.4
+       u0194.2
+       u0191.2
+       "u0191%b"  $  3
$   u01101%c
        u0155.2
+       u0125.2
+       u0123.2
+       u0122.2
+       u01101.2
+       "u01101%c"  $  5
$   u01101%d
        u0127.3
+       u01101.1
+       "u01101%d"  $  2
$   u01101%q
        u01102.1
+       u01101.4
+       "u01101%q"  $  2
$   u0129%cb
        u0137.3
+       u0129.7
+       "u0129%cb"  $  2
$   u0129%cn
        u0137.2
+       u0129.6
+       "u0129%cn"  $  2
$   u0129%cp
        u0137.1
+       u0129.5
+       "u0129%cp"  $  2
$   u01103%a
        u0142.2
+       u01103.1
+       "u01103%a"  $  2
$   u01103%b
        u0143.2
+       u01103.2
+       "u01103%b"  $  2
$   u01103%q
        u0156.2
+       u01103.4
+       "u01103%q"  $  2
$   u01103%s
        u0123.4
+       u01103.3
+       "u01103%s"  $  2
$   u0181%in1
        u0185.2
+       u0183.1
+       u0181.1
+       "u0181%in1"  $  3
$   u0147%q
        u015.2
+       u0147.2
```

```
+       "u0147%q"  $  2
$   u01104%in1
        u0194.3
+       u0125.4
+       u01104.1
+       "u01104%in1"  $  3
$   u0170%q
        u0174.1
+       u0172.1
+       u0170.2
+       "u0170%q"  $  3
$   u0122%q
        u0191.3
+       u0122.4
+       "u0122%q"  $  2
$   u0122%d
        u0129.3
+       u0122.1
+       "u0122%d"  $  2
$   u0171%q
        u0175.1
+       u0171.3
+       "u0171%q"  $  2
$   u0123%d
        u0128.3
+       u0123.1
+       "u0123%d"  $  2
$   u015%d
        u016.2
+       u015.1
+       u015.5
+       "u015%d"  $  3
$   u015%q
        u0194.1
+       u015.4
+       "u015%q"  $  2
$   out1
        u0189.2
+       out1  $  1
$   out2
        u01102.2
+       out2  $  1
$   out3
        u01104.2
+       out3  $  1
$   out4
        u0167.2
+       out4  $  1
$   u0199%d
        u0199.1
+       u0199.5
+       "u0199%d"  $  2
$   u0172%q
        u0176.1
+       u0172.3
+       "u0172%q"  $  2
$   u016%d
        u0199.2
+       u016.1
+       u016.5
+       "u016%d"  $  3
$   u016%q
        u0191.1
+       u016.4
+       "u016%q"  $  2
$   .gnd
        $ no connections
$   u0173%q
```

```
        u0177.1
+       u0173.2
+       "u0173%q"  $  2
$   u0125%d
        u0126.3
+       u0125.1
+       "u0125%d"  $  2
$   u0127%cb
        u0135.3
+       u0127.7
+       "u0127%cb"  $  2
$   u0127%cn
        u0135.2
+       u0127.6
+       "u0127%cn"  $  2
$   u0127%cp
        u0135.1
+       u0127.5
+       "u0127%cp"  $  2
$   u0174%q
        u0178.1
+       u0174.2
+       "u0174%q"  $  2
$   clk
        u0147.1
+       clk  $  1
$   u0175%q
        u0186.1
+       u0175.2
+       "u0175%q"  $  2
$   u0176%q
        u0187.1
+       u0176.2
+       "u0176%q"  $  2
$   u0177%q
        u0177.3
+       "u0177%q"  $  1
$   u0178%q
        u0178.3
+       "u0178%q"  $  1
$   in1
        u0142.1
+       in1  $  1
$   in2
        u0143.1
+       in2  $  1
$   in3
        u0169.1
+       in3  $  1
$   in4
        u0170.1
+       in4  $  1
$   in5
        u0185.1
+       in5  $  1
$   u0156%d
        u0157.2
+       u0156.1
+       u0156.5
+       "u0156%d"  $  3
$   u0157%d
        u0158.2
+       u0157.1
+       u0157.5
+       "u0157%d"  $  3
$   u0158%d
        u0159.2
+       u0158.1
```

```
+       u0158.5
+       "ù0158%d"  $  3

EXTERNALS $
        RES (IN) $
        IN1 (IN) $
        IN2 (IN) $
        IN3 (IN) $
        IN4 (IN) $
        IN5 (IN) $
        TEST (IN) $
        WR (IN) $
        CS (IN) $
        CLK (IN) $
        B0I (IN) $
        B1I (IN) $
        B2I (IN) $
        B3I (IN) $
        B0O (OUT) $
        B1O (OUT) $
        B2O (OUT) $
        B3O (OUT) $
        OUT1 (OUT) $
        OUT2 (OUT) $
        OUT3 (OUT) $
        OUT4 (OUT) $
SPECIAL INFO $

    NOFAULTS $

    TIMING $

    TECHNOLOGY $

END $
```

INTELLIGEN FAULT LOG SUMMARY
CIRCUIT "ser2par"

SUMMARY OF COMPLETION INFORMATION FOR ALL BUILDS ATTEMPTED:

COMPLETION STATUS GROUP	NUMBER OF BUILDS IN THIS GROUP	FRACTION OF TOTAL BUILDS IN THIS GROUP	ACTUAL NUMBER OF FAULTS REPRESENTED IN THIS GROUP
NO_TEST	29	0.5577	45
NO_PATH	8	0.1538	12
NO_DETECT	6	0.1154	6
OK	9	0.1731	10
TOTAL BUILDS:	52		

TOTAL UNTESTABLE FAULTS (NO_PATH + NO_EXERCI): 12

STATISTICS FOR TESTS BUILT (OK, NO_DETECT, NO_SOLID):

AVERAGE NUMBER OF VECS: 42
MAXIMUM VECTORS: 528
MAXIMUM BACKUPS: 23

AVERAGE ATG TIME: 9s
MAXIMUM ATG TIME: 122s
STANDARD DEVIATION: 31.05

DETECTION STATUS WHEN THIS REPORT WAS CREATED:

TOTAL NUMBER OF FAULTS SELECTED: 180
TOTAL CLASSES: 153
FAULTS PER PASS: 1500
INDEX IN FAULT LIST OF LAST BUILD: 95
INDEX OF NEXT PASS BOUNDARY: 153
COMPLETION: 1.0000

NUMBER OF FAULTS NOT DETECTED: 61
NUMBER OF OSCILLATION DETECTS: 0
NUMBER OF POTENTIAL DETECTS: 4
NUMBER OF SOLID DETECTS: 115
TOTAL DETECTS: 119

ESTIMATED DETECTION:

$$\frac{\text{TOTAL DETECTS}}{(\text{TOTAL FAULTS} * \text{COMPLETION}) - \text{UNTESTABLE}} = \%70.83$$

=== INTELLIGEN FAULT LOG ===
CIRCUIT: "ser2par"
Wed Sep 20 21:01:53 1989
52 ENTRIES

DETECT	CYC	A+S	STATUS	B,V	ID	FAULT NAME
17.77	5	1+3	OK	0,2	53+	U0127(U4.Y)/1
28.33	534	123+9	OK	23,528	88+	U0158(U1.QN)/1
28.33	534	123+9	NO_TEST	20,1	74+	U016(U1.QN)/1
32.22	535	124+9	OK	0,1	56-	U0129(U3.Y)/0
34.44	536	124+10	OK	0,1	61+	U0123(U1.Q)/1
34.44	536	124+10	NO_TEST	18,1	66+	U0122(U1.Q)/1
38.88	543	125+10	NO_DETECT	0,5	63+	U0147(U1.Y)/1
46.11	547	126+11	OK	0,2	29-	U0110(U1.Q)/0
46.11	547	126+11	NO_TEST	1,1	23+	U0182(U1.Y)/1
46.11	547	126+11	NO_TEST	18,1	55-	U0137(U4.Y)/0
46.11	547	126+11	NO_TEST	3,1	11-	IN3/0
53.88	560	128+12	OK	1,9	51-	U0199(U1.Q)/0
58.33	564	128+13	OK	0,2	59+	U0143(U1.Y)/1
58.33	564	129+13	NO_TEST	18,1	67+	U0129(U4.Y)/1
58.33	564	129+13	NO_TEST	1,1	17+	U0167(U1.Y)/1
58.33	564	130+13	NO_TEST	26,1	75+	U016(U1.Q)/1
58.33	564	130+13	NO_TEST	2,1	13-	IN5/0
58.33	564	130+13	NO_TEST	6,1	49+	U0191(U1.Y0)/1
60.00	583	132+13	NO_DETECT	8,16	72+	U0199(U1.QN)/1
60.00	583	132+13	NO_TEST	1,1	48+	U0190(U1.Y)/1
60.00	583	132+13	NO_PATH	0,1	85+	U0178(U1.Y)/1
60.00	583	132+13	NO_TEST	2,1	62+	U0185(U1.Y)/1
63.88	587	133+13	OK	0,2	78-	U0135(U4.Y)/0
65.00	591	133+14	NO_DETECT	0,2	45+	U0126(U1.Y)/1
65.00	591	133+14	NO_TEST	3,1	12+	IN4/1
65.00	618	136+14	NO_DETECT	4,22	80+	U0127(U1.Y)/1
65.00	618	136+14	NO_TEST	18,1	55+	U0137(U4.Y)/1
65.00	618	137+14	NO_TEST	3,1	12-	IN4/0
65.00	647	140+15	NO_DETECT	4,22	27+	U0128(U1.Y)/1
65.00	647	140+15	NO_TEST	1,1	19+	TEST/1
65.00	647	140+15	NO_TEST	1,1	14+	U0189(U1.Y)/1
65.00	647	140+15	NO_TEST	2,1	13+	IN5/1
65.00	677	143+16	NO_DETECT	4,22	57+	U0129(U1.Y)/1
65.00	677	144+16	NO_TEST	14,1	66-	U0122(U1.Q)/0
65.00	677	144+16	NO_TEST	3,1	65+	U0170(U1.Y)/1
65.00	677	144+16	NO_PATH	0,1	76+	U0173(U1.Y)/1
65.00	677	144+16	NO_TEST	2,1	34+	U0187(U1.Y)/1
66.11	681	144+16	OK	0,2	59-	U0143(U1.Y)/0
66.11	681	144+16	NO_TEST	2,1	32+	U0186(U1.Y)/1
66.11	681	145+16	NO_TEST	6,1	49-	U0191(U1.Y0)/0
66.11	681	145+16	NO_TEST	20,1	75-	U016(U1.Q)/0
66.11	681	145+16	NO_PATH	0,1	28+	U0184(U1.Y)/1
66.11	681	145+16	NO_TEST	3,1	11+	IN3/1
66.11	681	146+16	NO_TEST	3,1	50-	U0169(U1.Y)/0
66.11	681	146+16	NO_PATH	0,1	28-	U0184(U1.Y)/0
66.11	681	146+16	NO_PATH	0,1	85-	U0178(U1.Y)/0
66.11	681	146+16	NO_PATH	0,1	84+	U0177(U1.Y)/1
66.11	681	146+16	NO_TEST	3,1	50+	U0169(U1.Y)/1
66.11	681	146+16	NO_TEST	3,1	65-	U0170(U1.Y)/0
66.11	681	146+16	NO_PATH	0,1	84-	U0177(U1.Y)/0
66.11	681	146+16	NO_PATH	0,1	81+	U0174(U1.Y)/1
66.11	681	146+16	NO_TEST	2,1	62-	U0185(U1.Y)/0

```
.PATTERN  1000  B0 B1 B2 B3 CLK CS IN1 IN2 DUMMY1 DUMMY2 DUMMY3 RES TEST WR in3
+         in5 in4

$         CREATED BY INTELLIGEN  Wed Sep 20 21:07:27 1989

$         For circuit ser2par


$         Zero Vector
          00000000000010000
          00000000000010000
          00000000000010001
          00000000000010000
          00000000000010001
          00000000000010011
          00000000000010110
          00000000000010111

$         U0127(U4.Y)/1
          00000000000010011
          00000000000010000
          00000000000010001
          00000000000010000
          00000000000010001
.
.
.
.
          0zzz1011000111011
          0zzz1011000111010
          0zzz1011000111011
          0zzz1011000111010
          0zzz1011000111011
          0zzz1011000111011
          0zzz1011000111110
          0zzz1011000111111

$         5448 total cycles
.EOP
```

```
.SLOW

+ u01103%b
+ u0122%d
+ u0122%q
+ u016%q
+ u0168%a
+ u0171%q
+ u0172%q
+ u0175%q
+ u0176%q
+ u0181%q
+ u0182%q
+ u0186%q
+ u0187%q


.SHIGH

+ out1
+ te
+ u0122%d
+ u0122%q
+ u016%q
+ u0168%a
+ u0168%s
+ u0171%q
+ u0172%q
+ u0175%q
+ u0176%q
+ u0181%q
+ u0182%q
+ u0186%q
+ u0187%q
+ u0188%q
+ u0199%d
```

```
+ u01103.2 /0 $
+ u0122.1 /0 $
+ u0122.4 /0 $
+ u016.4 /0 $
+ u0168.1 /0 $
+ u0171.3 /0 $
+ u0172.3 /0 $
+ u0175.2 /0 $
+ u0176.2 /0 $
+ u0181.3 /0 $
+ u0182.2 /0 $
+ u0186.2 /0 $
+ u0187.2 /0 $
+ tctrl1.3 /1 $
+ u0122.1 /1 $
+ u0122.4 /1 $
+ u016.4 /1 $
+ u0168.1 /1 $
+ u0168.3 /1 $
+ u0171.3 /1 $
+ u0172.3 /1 $
+ u0175.2 /1 $
+ u0176.2 /1 $
+ u0181.3 /1 $
+ u0182.2 /1 $
+ u0186.2 /1 $
+ u0187.2 /1 $
+ u0188.4 /1 $
+ u0199.1 /1 $
```

INTELLIGEN FAULT LOG SUMMARY
CIRCUIT "ser2par"

SUMMARY OF COMPLETION INFORMATION FOR ALL BUILDS ATTEMPTED:

COMPLETION STATUS GROUP	NUMBER OF BUILDS IN THIS GROUP	FRACTION OF TOTAL BUILDS IN THIS GROUP	ACTUAL NUMBER OF FAULTS REPRESENTED IN THIS GROUP
OK	8	1.0000	20
TOTAL BUILDS:	8		

TOTAL UNTESTABLE FAULTS (NO_PATH + NO_EXERCI): 0

STATISTICS FOR TESTS BUILT (OK, NO_DETECT, NO_SOLID):

AVERAGE NUMBER OF VECS: 4
MAXIMUM VECTORS: 16
MAXIMUM BACKUPS: 2

AVERAGE ATG TIME: 1s
MAXIMUM ATG TIME: 2s
STANDARD DEVIATION: 0.00

DETECTION STATUS WHEN THIS REPORT WAS CREATED:

TOTAL NUMBER OF FAULTS SELECTED: 29
TOTAL CLASSES: 16
FAULTS PER PASS: 1500
INDEX IN FAULT LIST OF LAST BUILD: 15
INDEX OF NEXT PASS BOUNDARY: 16
COMPLETION: 1.0000

NUMBER OF FAULTS NOT DETECTED: 0
NUMBER OF OSCILLATION DETECTS: 0
NUMBER OF POTENTIAL DETECTS: 0
NUMBER OF SOLID DETECTS: 29
TOTAL DETECTS: 29

ESTIMATED DETECTION:

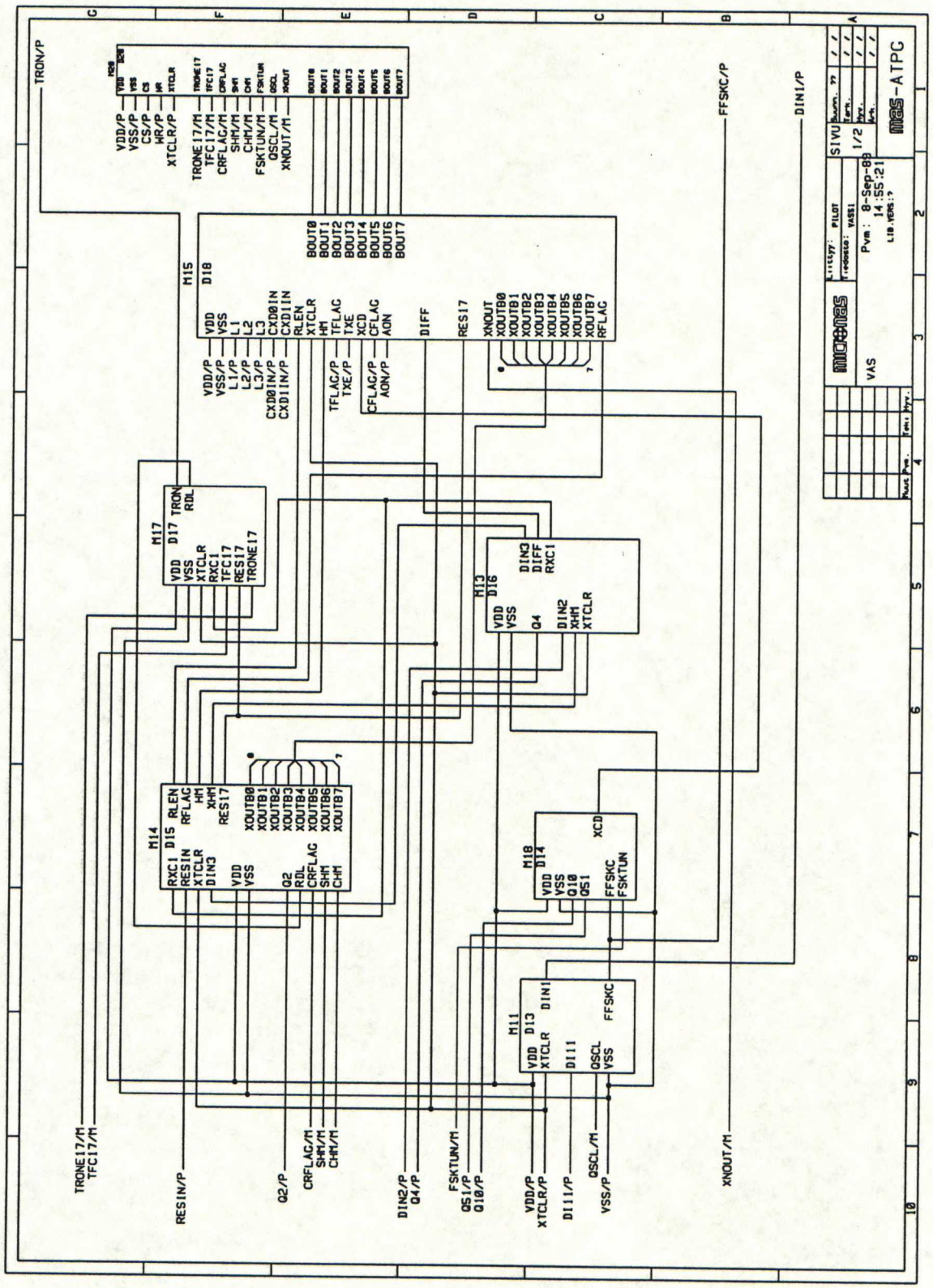
$$\frac{\text{TOTAL DETECTS}}{(\text{TOTAL FAULTS} * \text{COMPLETION}) - \text{UNTESTABLE}} = \%100.00$$

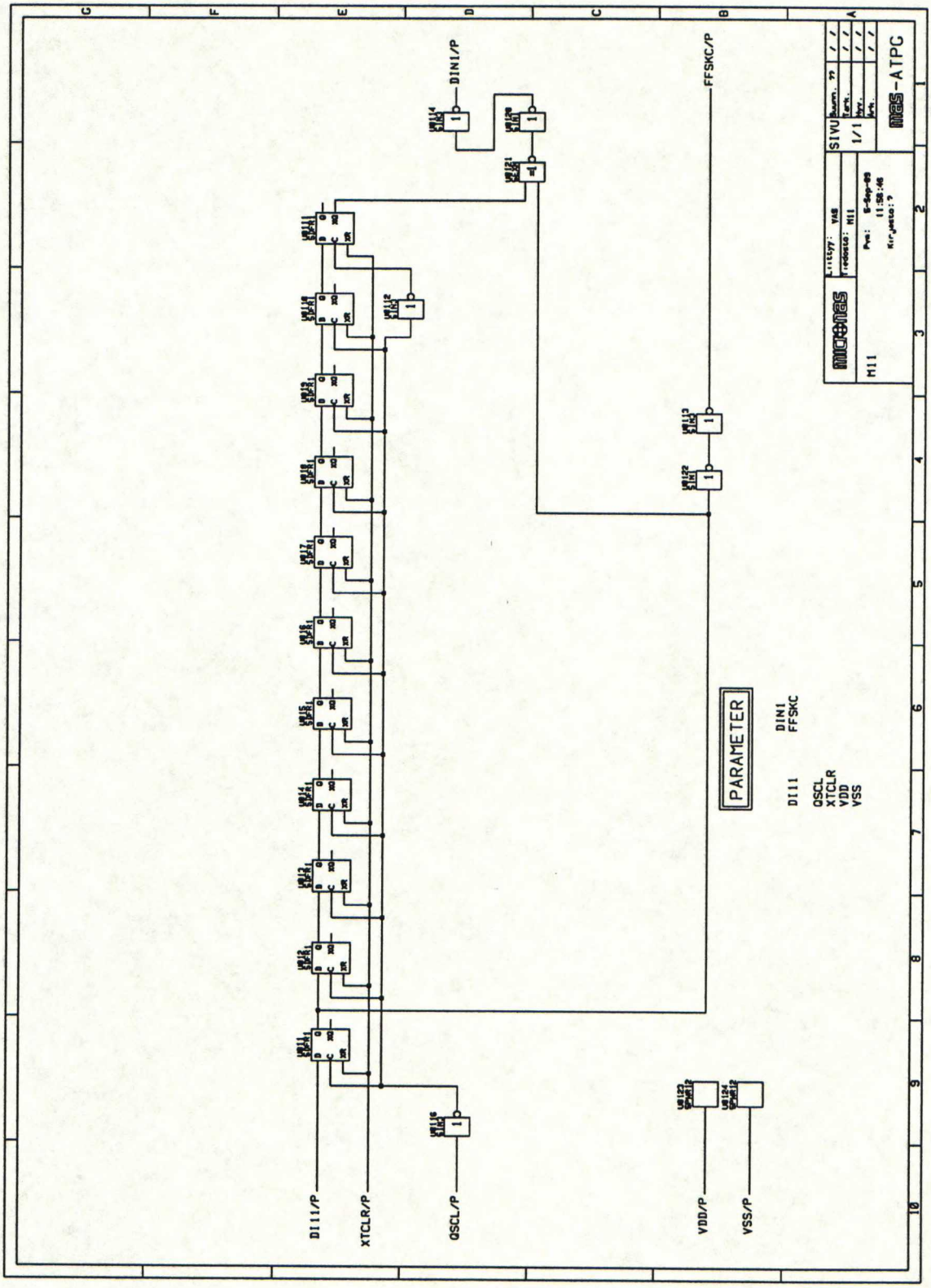
=== INTELLIGEN FAULT LOG ===
CIRCUIT: "ser2par"
Thu Sep 21 07:57:46 1989
8 ENTRIES

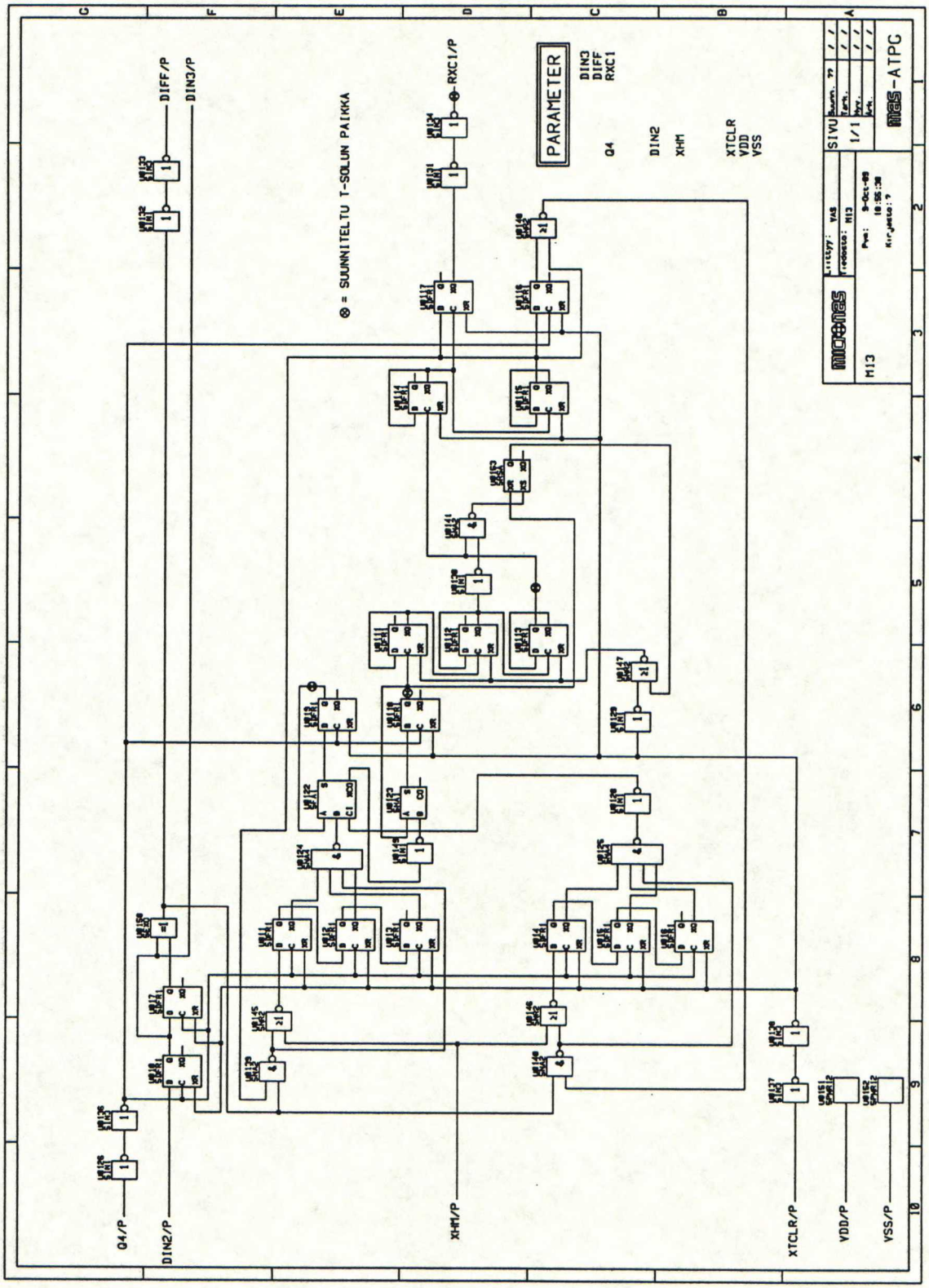
DETECT	CYC	A+S	STATUS	B,V	ID	FAULT NAME
24.13	7	1+1	OK	1,4	60-	U0137(U4.Y)/0
34.48	8	1+2	OK	0,1	33+	U0187(U1.Y)/1
48.27	14	2+2	OK	0,4	80-	U016(U1.Q)/0
79.31	15	2+2	OK	0,1	14+	U0189(U1.Y)/1
89.65	16	2+2	OK	0,1	31+	U0186(U1.Y)/1
93.10	23	3+2	OK	0,5	64-	U0143(U1.Y)/0
96.55	41	5+3	OK	2,16	76+	U0199(U1.QN)/1
100.00	49	6+3	OK	1,6	41+	TCTRL1(TE.Y)/1

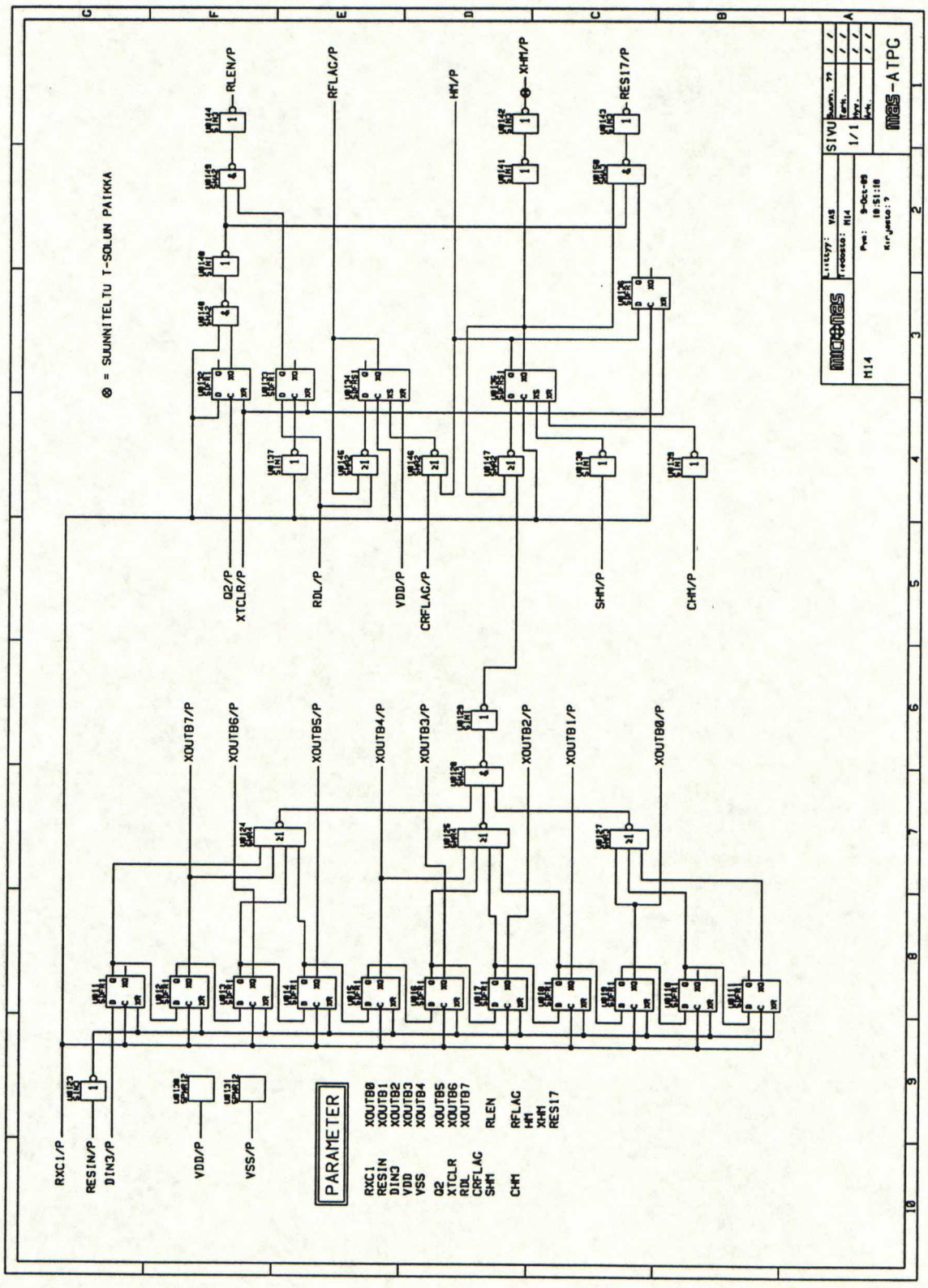
```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  ser2par.cmd  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                                 O U T P U T S
$
$  S I L O S  I I   ver 1.005        Thu Sep 21 13:13:02 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

              rwcct iiiii bbbb oooo
              ersle nnnnn 0123 uuuu
              s  ks 12345      tttt
                 t             1234

        TIME
         900  00001 00000 0000 100?
        1900  00001 00000 0000 100?
        2900  00001 00010 0000 100*
        3900  00001 00000 0000 100*
        4900  00001 00010 0000 1000
        5900  00001 00011 0000 1000
        6900  00001 00101 0000 1000
.
.
.
.
     5494900  11011 11111 0ZZZ 1100
     5495900  11011 11111 0ZZZ 1100
     5496900  11011 11111 0ZZZ 1100
     5497900  11011 11111 0ZZZ 1100
     5498900  11011 11111 0ZZZ 1100
     5499900  11011 11111 0ZZZ 1100
         900  00000 00000 0000 1000
        1900  00000 00000 0000 1000
        2900  10000 00000 0000 1000
        3900  10010 00000 0000 1000
        4900  10000 00000 0001 1000
        5900  11000 00000 0001 1000
        6900  11010 00000 0001 1000
.
.
.
.
       54900  11110 11011 0011 1000
       55900  11110 11011 0011 1000
       56900  11110 11011 0011 1000
       57900  11110 11011 0011 1000
       58900  11110 11011 0011 1000
       59900  11110 11011 0011 1000
```

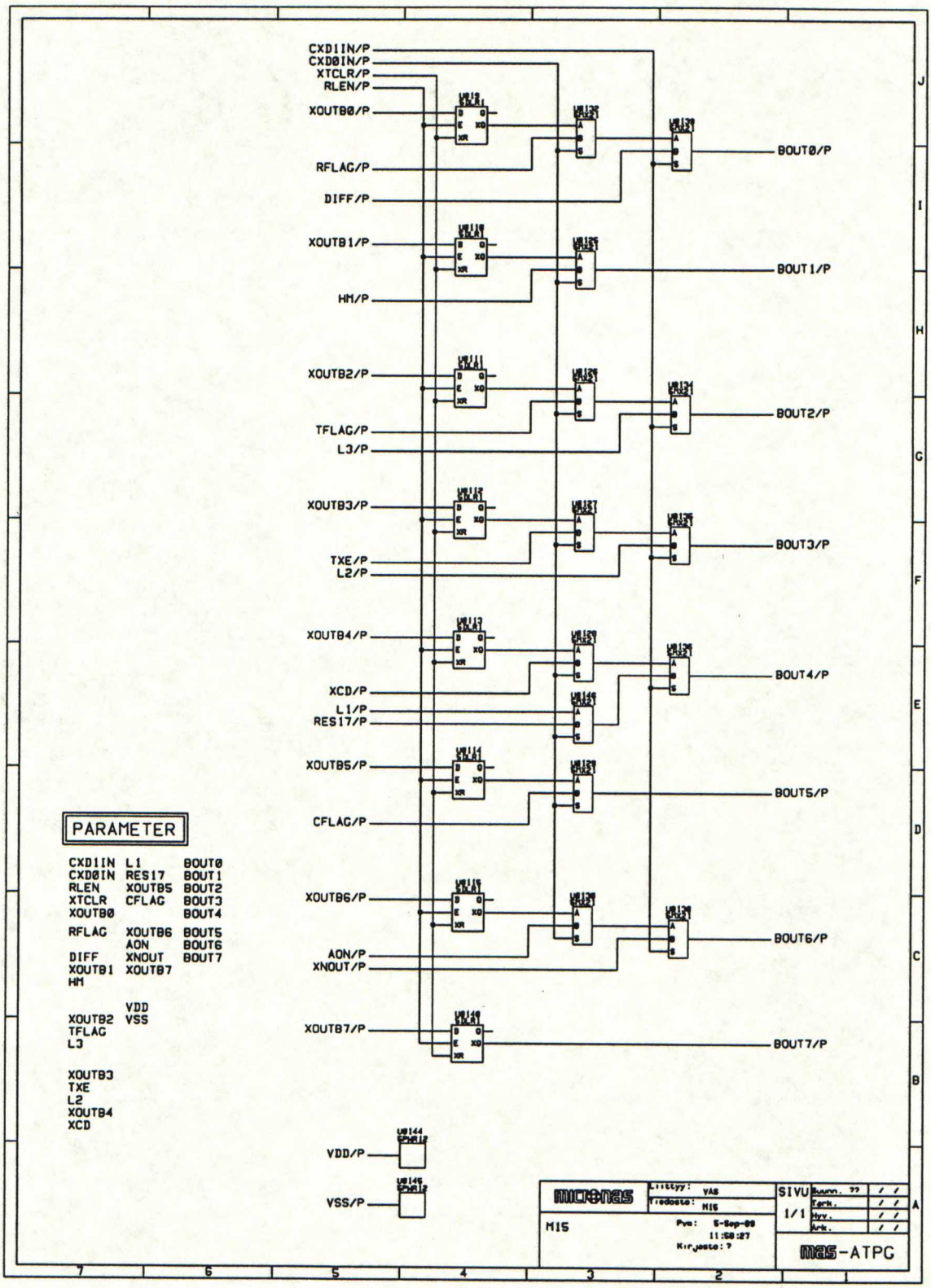
```
PATTERN
XXXXXX,HSDM PATTERN
(THIS,IS,THE,CHANNEL,LIST, .....)
;
;*TERAFORM FINAL FORMAT PRODUCER*  HAS PROCESSED THIS DATA AT 13:24 21-SEP-1989
;
;***** THE  LOOP COMPRESSOR  ***** HAS PROCESSED THIS DATA AT 13:24 21-SEP-1989
;
;***** THE  SKIP COMPRESSOR  ***** HAS PROCESSED THIS DATA AT 13:22 21-SEP-1989
;
;*****  SILTOTSE TRANSLATOR  ***** HAS PROCESSED THIS DATA AT 13:22 21-SEP-1989
;
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  ser2par.cmd  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;$
;$                                O U T P U T S
;$
;$  S I L O S  I I    ver 1.005         Thu Sep 21 13:13:02 1989
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;
;
;             rwcct iiiii bbbb oooo
;             ersle nnnnn 0123 uuuu
;             s  ks 12345      tttt
;                t             1234
;
;       TIME
DR  LLLLH LLLLL LLLL OOOO
EX  LLLLH LLLLL LLLL HLLH
CA  NNNNN NNNNN NNNN YYYN                                SKIP=  1*;          0
HI  LLLLH LLLHL LLLL HLLH                                SKIP=  0*;          1
HI  LLLLH LLLLL LLLL HLLH                                SKIP=  0*;          2
DR  LLLLH LLLHL LLLL OOOO
EX  LLLLH LLLHL LLLL HLLL
CA  NNNNN NNNNN NNNN YYYY                                SKIP=  0*;          3
HI  LLLLH LLLHH LLLL HLLL                                SKIP=  0*;          4
HI  LLLLH LLHLH LLLL HLLL                                SKIP=  0*;          5
.
.
.
.
HI  HHLHH HHLLH LLHH HLLL                                SKIP=  0*;       2601
HI  HHLHH HHLHH LLHH HLLH                                SKIP=  0*;       2602
HI  LLLLL HHLHH LLHH HLLL                                SKIP=  0*;       2603
HI  HLLLL HHLHH LLHH HLLL                                SKIP=  0*;       2604
HI  HHLLL HHLHH LLHH HLLL                                SKIP=  0*;       2605
DR  HHHHL HHLHH OOOO OOOO
EX  HHHHL HHLHH LLHH HLLL
CA  NNNNN NNNNN YYYY YYYY                                SKIP= 11 ;       2606
PATEND *
END
```

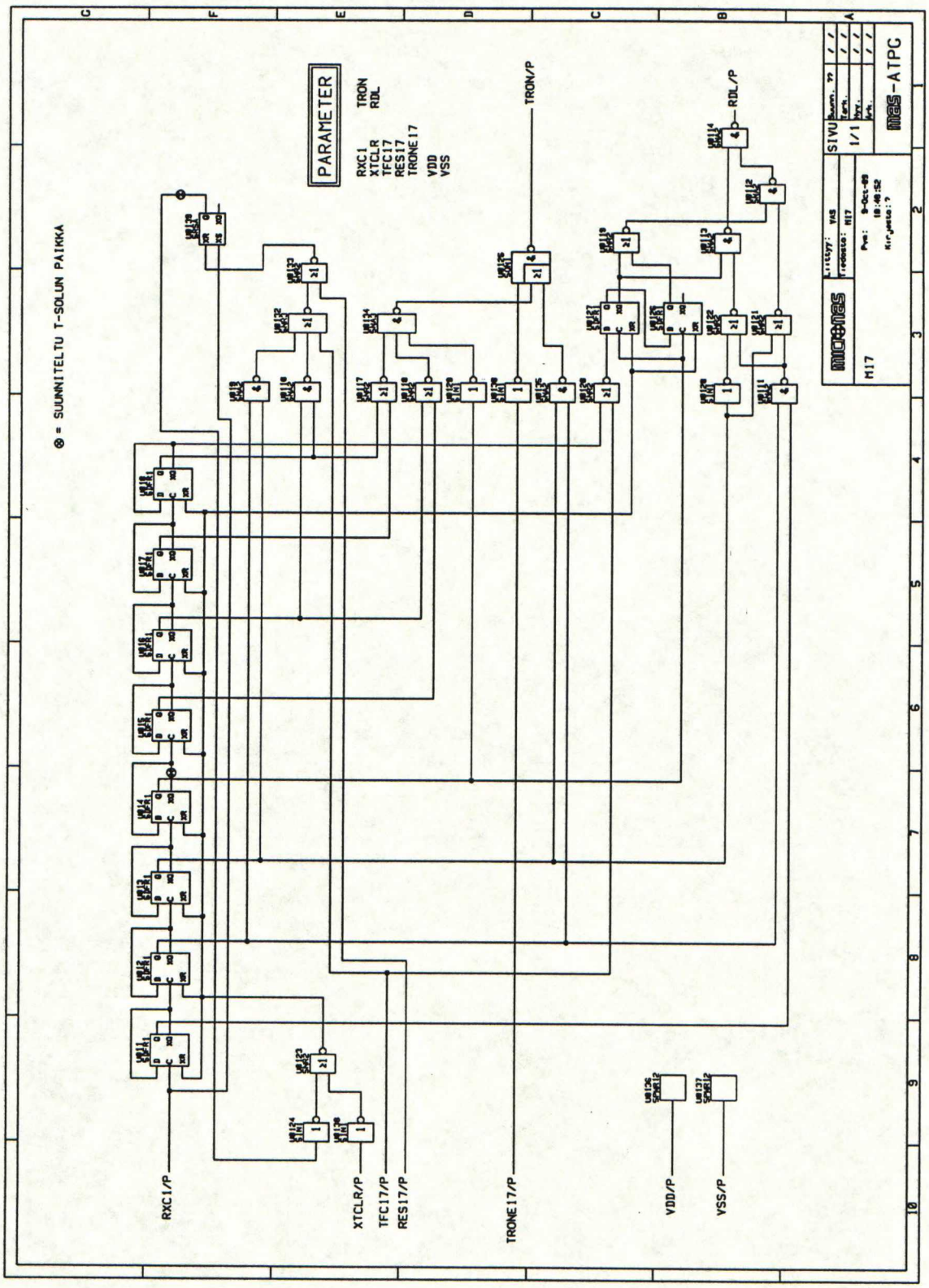

TRON/P
FFSKC/P
DIN1/P
VDD/P
VSS/P
CS/P
WR/P
XTCLR/P
TRONE17/M
TFC17/M
CRFLAG/M
SHM/M
CHM/M
FSKTUN/M
QSCL/M
XNOUT/M
BOUT0
BOUT1
BOUT2
BOUT3
BOUT4
BOUT5
BOUT6
BOUT7
M15 D18
VDD
VSS
L1
L2
L3
CXD0IN
CXD1IN
RLEN
XTCLR
HM
TFLAG
TXE
XCD
CFLAG
AON
DIFF
RES17
XNOUT
XOUTB0
XOUTB1
XOUTB2
XOUTB3
XOUTB4
XOUTB5
XOUTB6
XOUTB7
RFLAG
VDD/P
VSS/P
L1/P
L2/P
L3/P
CXD0IN/P
CXD1IN/P
TFLAG/P
TXE/P
CFLAG/P
AON/P
M17 D17
TRON
RDL
VDD
VSS
XTCLR
RXC1
TFC17
RES17
TRONE17
M13 D16
VDD
VSS
Q4
DIN3
DIFF
RXC1
DIN2
XHM
XTCLR
M14 D15
RXC1
RESIN
XTCLR
DIN3
RLEN
RFLAG
HM
XHM
RES17
XOUTB0
XOUTB1
XOUTB2
XOUTB3
XOUTB4
XOUTB5
XOUTB6
XOUTB7
VDD
VSS
Q2
RDL
CRFLAG
SHM
CHM
M18 D14
VDD
VSS
Q10
QS1
XCD
FFSKC
FSKTUN
M11 D13
VDD
XTCLR
DIN1
D111
QSCL
VSS
FFSKC
TRONE17/M
TFC17/M
RESIN/P
Q2/P
CRFLAG/M
SHM/M
CHM/M
DIN2/P
Q4/P
FSKTUN/M
QS1/P
Q10/P
VDD/P
XTCLR/P
D111/P
QSCL/M
VSS/P
XNOUT/M
SIVU
1/2
Pvm: 8-Sep-88
14:55:21
LIB.VERS:?
mas-ATPC

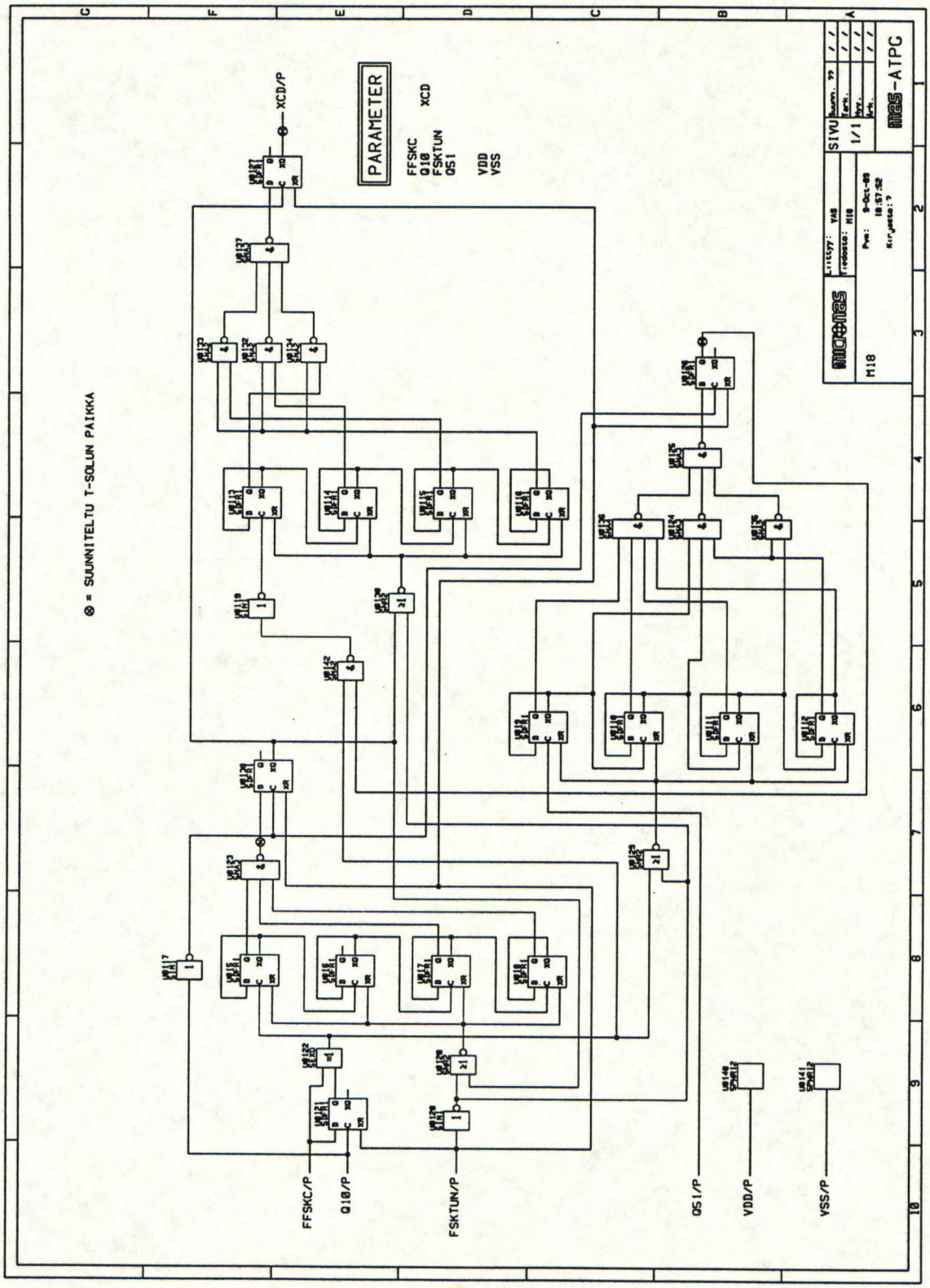

DIN1/P
FFSKC/P
PARAMETER
DI11
DIN1
FFSKC
QSCL
XTCLR
VDD
VSS
DI11/P
XTCLR/P
QSCL/P
VDD/P
VSS/P
mas-ATPC
SIVU
1/1
H11

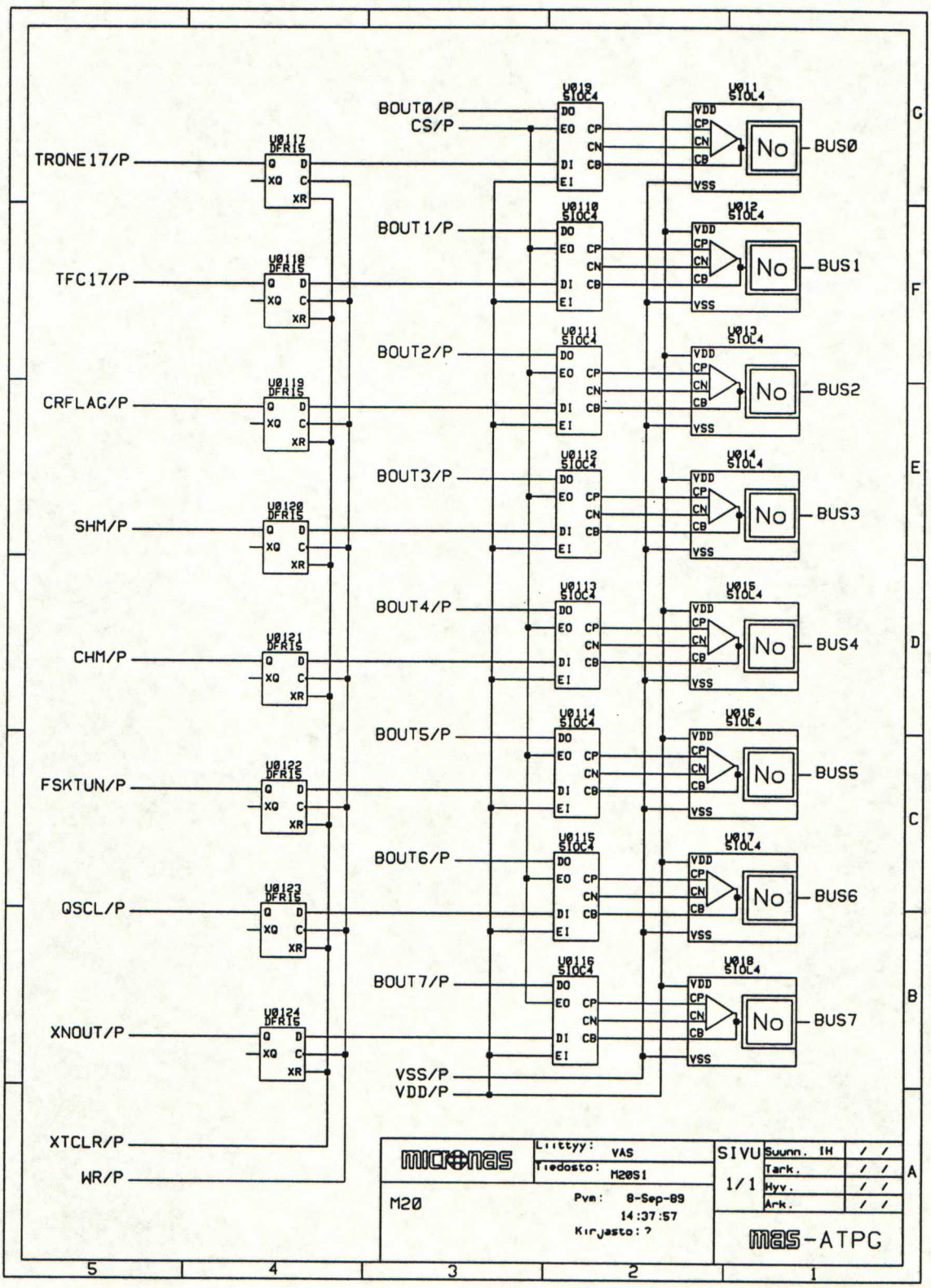

⊗ = SUUNNITELTU T-SOLUN PAIKKA
PARAMETER
DIN3
DIFF
RXC1
Q4
DIN2
XHM
XTCLR
VDD
VSS
DIFF/P
DIN3/P
RXC1/P
Q4/P
DIN2/P
XHM/P
XTCLR/P
VDD/P
VSS/P
mas-ATPC


⊗ = SUUNNITELTU T-SOLUN PAIKKA
PARAMETER
RXC1 XOUTB0
RESIN XOUTB1
DIN3 XOUTB2
VDD XOUTB3
VSS XOUTB4
Q2 XOUTB5
XTCLR XOUTB6
RDL XOUTB7
CRFLAG RLEN
SHM RFLAG
CHM HM
XHM
RES17
RXC1/P
RESIN/P
DIN3/P
VDD/P
VSS/P
Q2/P
XTCLR/P
RDL/P
VDD/P
CRFLAG/P
SHM/P
CHM/P
RLEN/P
RFLAG/P
HM/P
XHM/P
RES17/P
XOUTB7/P
XOUTB6/P
XOUTB5/P
XOUTB4/P
XOUTB3/P
XOUTB2/P
XOUTB1/P
XOUTB0/P
M14
1/1
mas-ATPC


CXD1IN/P
CXD0IN/P
XTCLR/P
RLEN/P
XOUTB0/P
RFLAG/P
DIFF/P
BOUT0/P
XOUTB1/P
HM/P
BOUT1/P
XOUTB2/P
TFLAG/P
L3/P
BOUT2/P
XOUTB3/P
TXE/P
L2/P
BOUT3/P
XOUTB4/P
XCD/P
L1/P
RES17/P
BOUT4/P
XOUTB5/P
CFLAG/P
BOUT5/P
XOUTB6/P
AON/P
XNOUT/P
BOUT6/P
XOUTB7/P
BOUT7/P
VDD/P
VSS/P
PARAMETER
CXD1IN L1 BOUT0
CXD0IN RES17 BOUT1
RLEN XOUTB5 BOUT2
XTCLR CFLAG BOUT3
XOUTB0 BOUT4
RFLAG XOUTB6 BOUT5
AON BOUT6
DIFF XNOUT BOUT7
XOUTB1 XOUTB7
HM
VDD
XOUTB2 VSS
TFLAG
L3
XOUTB3
TXE
L2
XOUTB4
XCD
MICRONAS
MIS
SIVU
1/1
mas-ATPG


⊗ = SUUNNITELTU T-SOLUN PAIKKA
PARAMETER
TRON
RDL
RXC1
XTCLR
TFC17
RES17
TRONE17
VDD
VSS
TRON/P
RDL/P
RXC1/P
XTCLR/P
TFC17/P
RES17/P
TRONE17/P
VDD/P
VSS/P
micronas
H17
SIVU
1/1
mas-ATPC


PARAMETER
FFSKC
Q10
FSKTUN
Q51
XCD
VDD
VSS
XCD/P
⊗ = SUUNNITELTU T-SOLUN PAIKKA
FFSKC/P
Q10/P
FSKTUN/P
Q51/P
VDD/P
VSS/P
SIVU
1/1
M18
mas-ATPC


BOUT0/P
CS/P
BOUT1/P
BOUT2/P
BOUT3/P
BOUT4/P
BOUT5/P
BOUT6/P
BOUT7/P
VSS/P
VDD/P
TRONE17/P
TFC17/P
CRFLAG/P
SHM/P
CHM/P
FSKTUN/P
QSCL/P
XNOUT/P
XTCLR/P
WR/P
BUS0
BUS1
BUS2
BUS3
BUS4
BUS5
BUS6
BUS7
No
micronas
M20
Liittyy: VAS
Tiedosto: M20S1
Pvm: 8-Sep-89
14:37:57
Kirjasto: ?
SIVU 1/1
Suunn. IH
Tark.
Hyv.
Ark.
mas-ATPG

INTELLIGEN FAULT LOG SUMMARY
CIRCUIT "vas"

SUMMARY OF COMPLETION INFORMATION FOR ALL BUILDS ATTEMPTED:

COMPLETION STATUS GROUP	NUMBER OF BUILDS IN THIS GROUP	FRACTION OF TOTAL BUILDS IN THIS GROUP	ACTUAL NUMBER OF FAULTS REPRESENTED IN THIS GROUP
ABANDONED	3	0.0508	3
ABORTED	2	0.0339	3
NO_DETECT	21	0.3559	23
OK	33	0.5593	36
TOTAL BUILDS:	59		

TOTAL UNTESTABLE FAULTS (NO_PATH + NO_EXERCI): 0

STATISTICS FOR TESTS BUILT (OK, NO_DETECT, NO_SOLID):

AVERAGE NUMBER OF VECS: 34
MAXIMUM VECTORS: 311
MAXIMUM BACKUPS: 242

AVERAGE ATG TIME: 8s
MAXIMUM ATG TIME: 89s
STANDARD DEVIATION: 17.44

DETECTION STATUS WHEN THIS REPORT WAS CREATED:

TOTAL NUMBER OF FAULTS SELECTED: 590
TOTAL CLASSES: 485
FAULTS PER PASS: 1500
INDEX IN FAULT LIST OF LAST BUILD: 454
INDEX OF NEXT PASS BOUNDARY: 485
COMPLETION: 1.0000

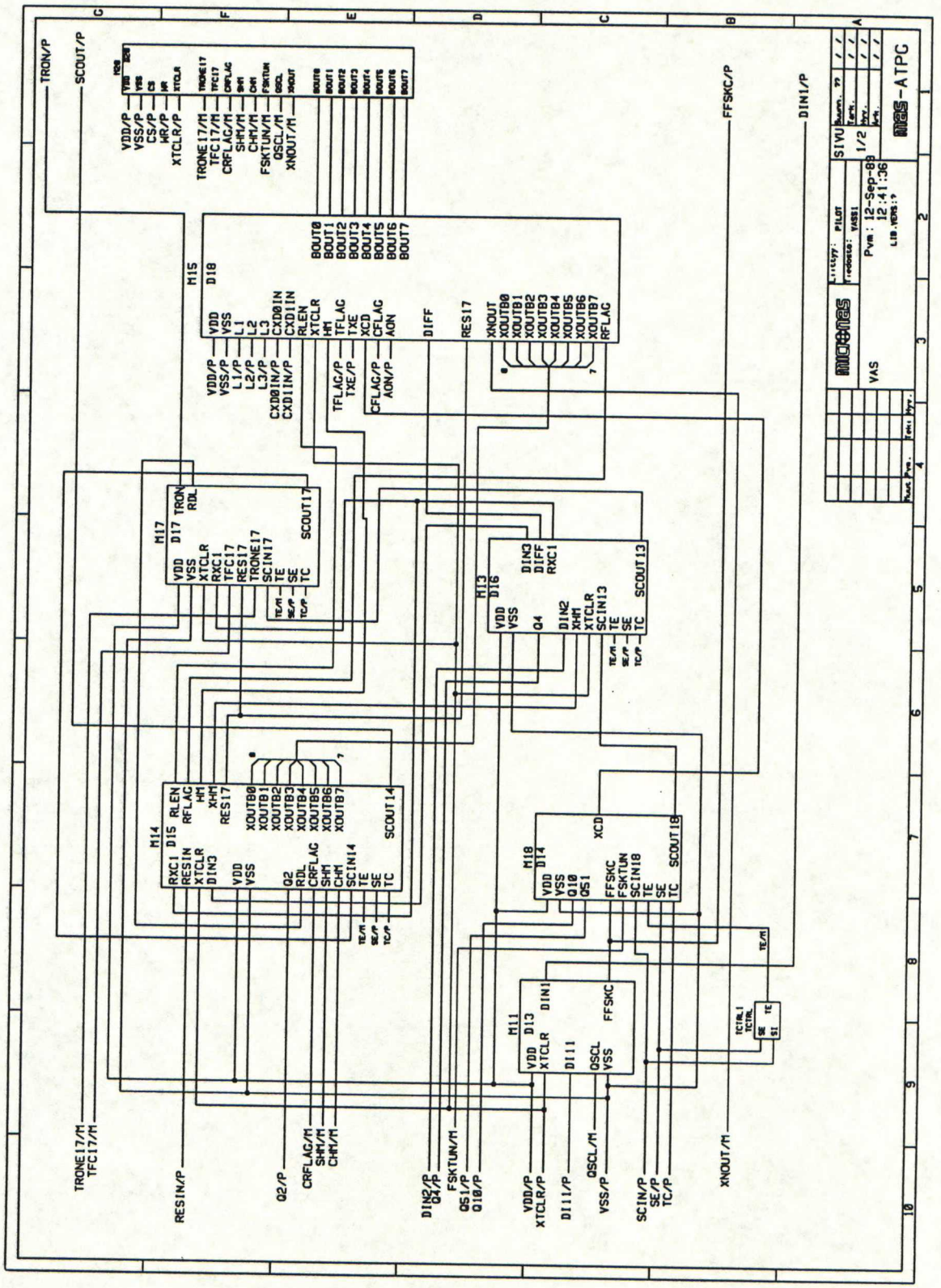
NUMBER OF FAULTS NOT DETECTED: 24
NUMBER OF OSCILLATION DETECTS: 0
NUMBER OF POTENTIAL DETECTS: 16
NUMBER OF SOLID DETECTS: 550
TOTAL DETECTS: 566

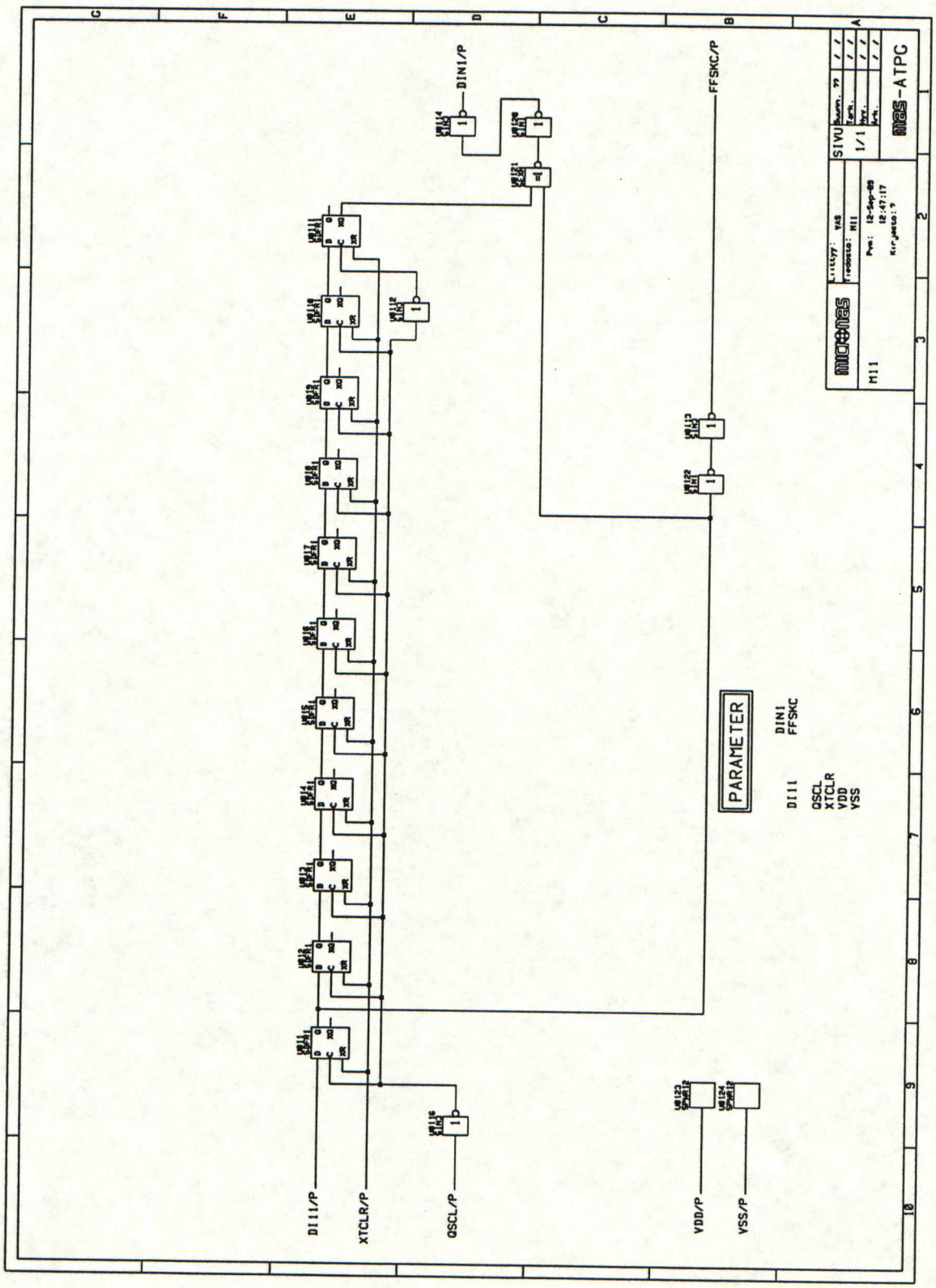
ESTIMATED DETECTION:

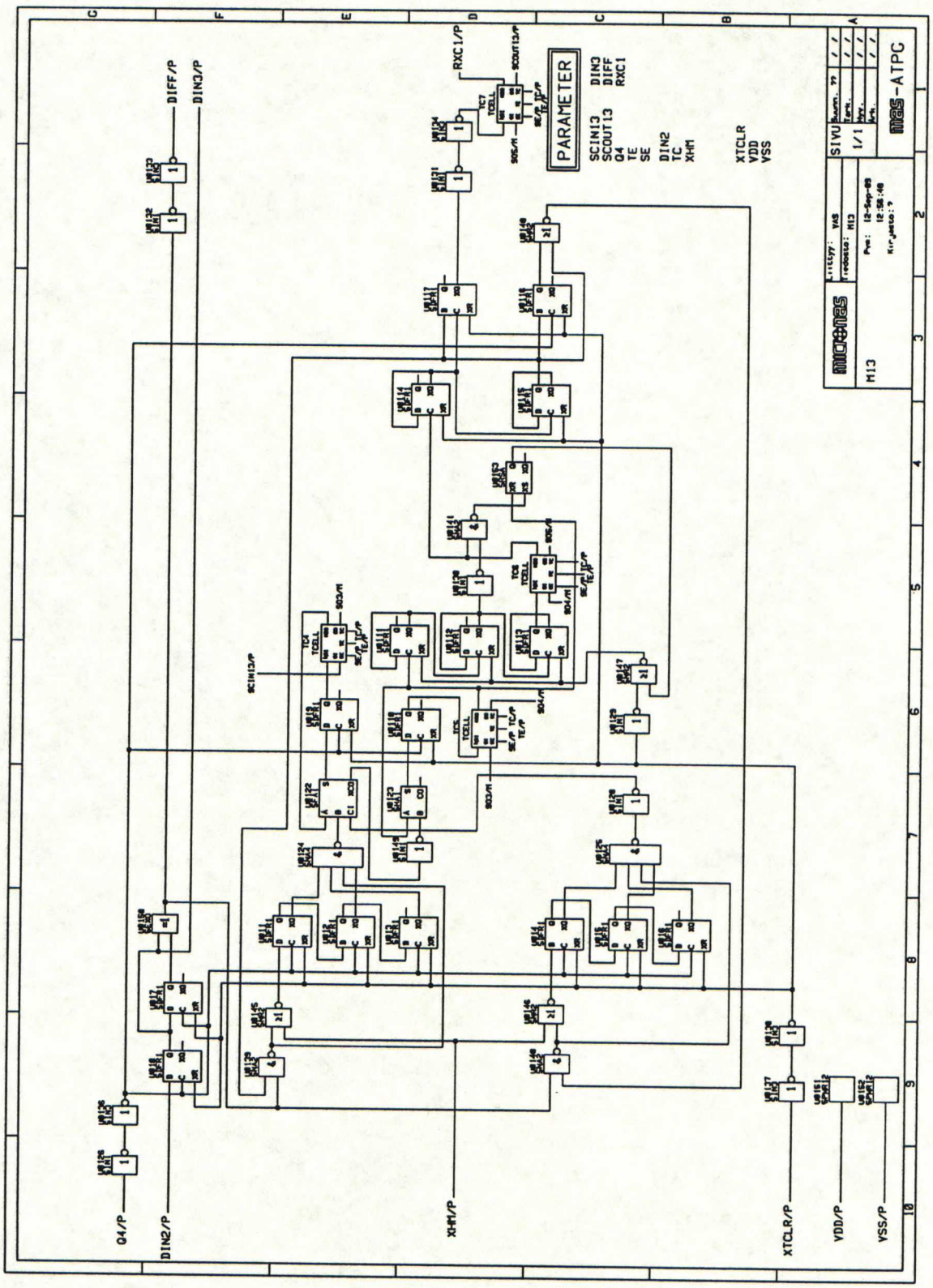
$$\frac{\text{TOTAL DETECTS}}{(\text{TOTAL FAULTS} * \text{COMPLETION}) - \text{UNTESTABLE}} = \%95.93$$

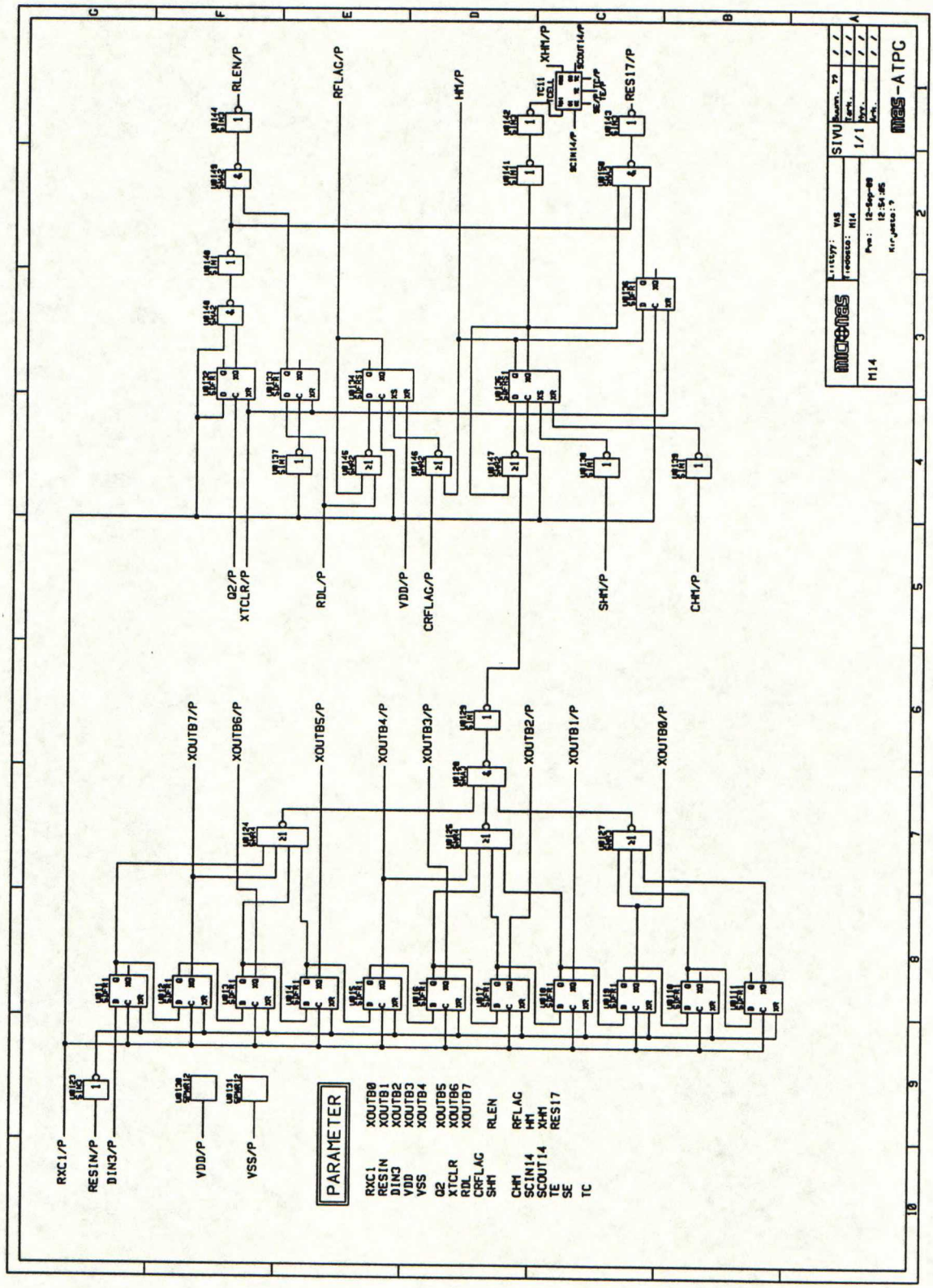
=== INTELLIGEN FAULT LOG ===
CIRCUIT: "vas"
Thu Sep 21 15:56:19 1989
59 ENTRIES

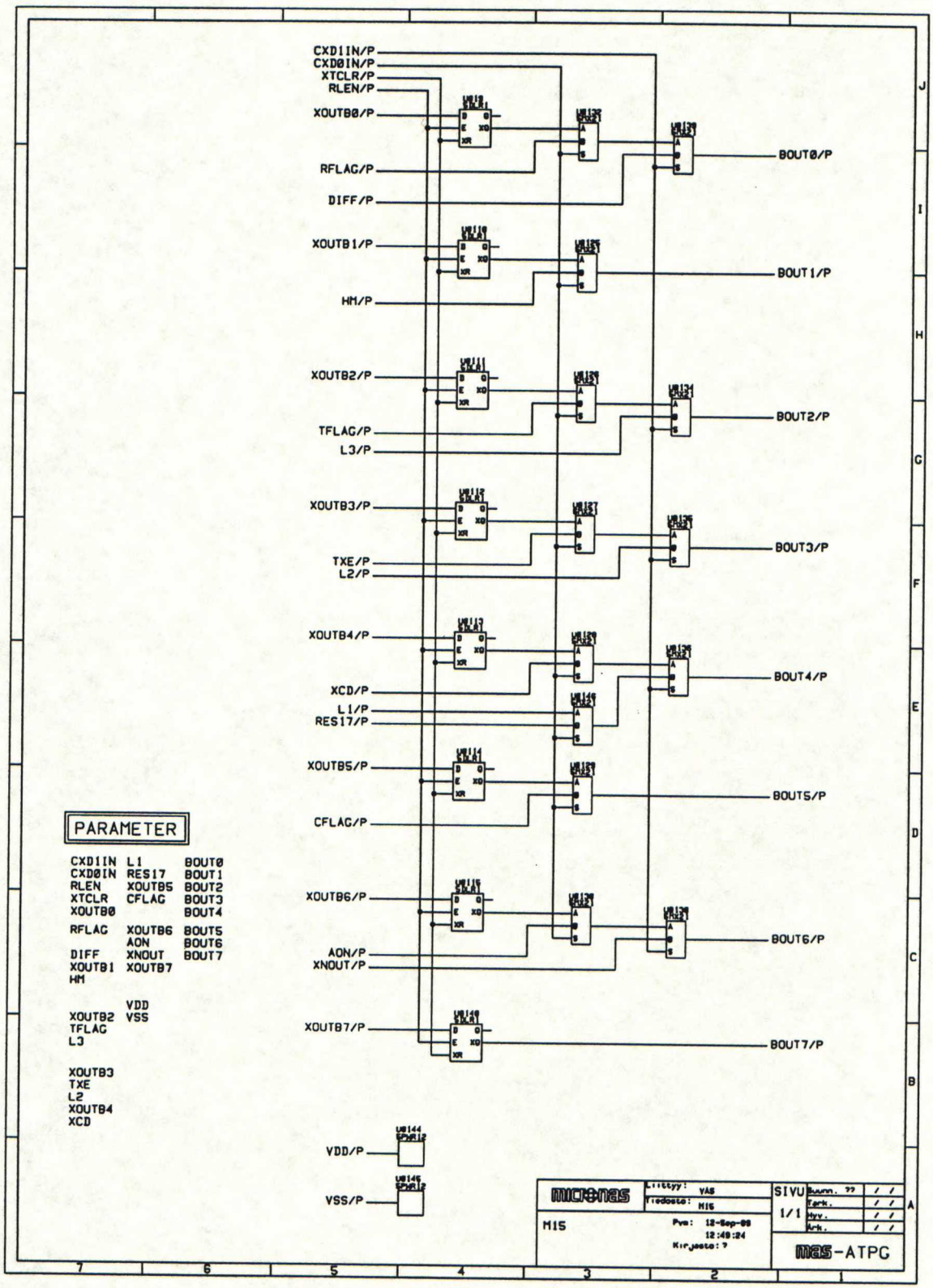
DETECT	CYC	A+S	STATUS	B,V	ID	FAULT NAME
18.13	42	9+11	OK	21,39	170-	U0120CAA(U1.Y)/0
21.18	43	9+13	OK	0,1	41-	U0136BAA(U1.Y0)/0
34.06	94	16+17	OK	1,34	259+	U0124FAA(U1.Y)/1
47.28	133	20+22	NO_DETECT	0,25	121-	U0110EAA(U1.Q)/0
50.00	134	20+24	OK	0,1	20+	L1/1
57.96	155	21+27	OK	0,15	181+	U017EAA(U1.QN)/1
60.84	161	22+30	OK	0,3	140+	U0113BAA(U1.QN)/1
60.84	161	31+30	ABORTED	3,2	237-	U0153DAA(U1.Q)/0
63.89	196	34+36	NO_DETECT	1,25	97-	U0134EAA(U1.QN)/0
68.81	299	54+39	OK	242,56	243+	U017FAA(U1.Q)/1
71.69	458	84+45	OK	8,156	134-	U0132CAA(U1.Y)/0
77.96	612	111+50	OK	22,90	90-	U0122FAA(U1.Y)/0
78.64	627	112+51	NO_DETECT	6,7	24+	Q2/1
79.15	628	113+51	OK	0,1	261+	U0110AAA(U3.Y)/1
81.01	645	115+52	OK	2,16	162+	U0113DAA(U1.QN)/1
84.40	661	117+53	OK	1,13	102+	U0114DAA(U1.QN)/1
84.74	662	117+53	OK	0,1	30-	TXE/0
85.25	667	118+54	NO_DETECT	0,4	267+	U019AAA(U1.Y)/1
85.76	690	119+55	OK	0,17	180+	U018EAA(U1.QN)/1
86.44	691	120+55	OK	0,1	44-	U0140BAA(U1.QN)/0
86.61	692	120+55	NO_DETECT	0,1	194+	U0114AAA(U1.Y)/1
86.77	698	120+55	OK	0,3	87-	U0116AAA(U3.Y)/0
86.77	699	121+56	NO_DETECT	0,1	135+	U0113AAA(U1.Y)/1
86.77	699	130+56	ABORTED	4,2	242+	U0141DAA(U1.Y)/1
87.79	790	146+59	OK	11,88	109-	U019CAA(U1.Y)/0
87.96	791	147+59	NO_DETECT	0,1	245+	U0115AAA(U1.Y)/1
89.15	979	180+63	NO_DETECT	63,106	279+	U0116FAA(U1.QN)/1
89.15	979	429+63	ABANDONED	2001,80	254+	U016AAA(U4.Y)/1
89.32	1010	432+65	OK	1,21	289-	U0111EAA(U1.QN)/0
89.49	1020	434+65	OK	2,9	231+	U0130DAA(U1.Y)/1
89.66	1021	434+65	NO_DETECT	0,1	70+	U0112AAA(U1.Y)/1
90.67	1035	436+66	OK	0,12	272-	U012DAA(U1.Q)/0
90.67	1035	685+66	ABANDONED	2001,80	100+	U0114AAA(U4.Y)/1
91.18	1040	685+66	OK	0,2	84-	U013AAA(U4.Y)/0
91.52	1041	686+66	OK	0,1	173-	U0146BAA(U1.Y0)/0
91.69	1042	686+67	OK	0,1	22-	L3/0
91.69	1091	692+68	NO_DETECT	3,26	161+	U0132FAA(U1.Y)/1
91.69	1092	692+68	NO_DETECT	0,1	287+	U0116AAA(U1.Y)/1
91.86	1129	696+70	OK	0,25	253+	U0124EAA(U1.Y)/1
92.37	1288	728+75	OK	8,156	149-	U0118AAA(U1.Q)/0
92.54	1323	731+76	OK	0,25	208+	U0125EAA(U1.Y)/1
92.54	1477	757+79	NO_DETECT	19,90	57+	U0134FAA(U1.Y)/1
92.71	1492	759+79	NO_DETECT	1,13	188+	U015DAA(U1.QN)/1
93.05	1496	760+80	OK	0,2	104+	U018AAA(U4.Y)/1
93.38	1497	760+80	OK	0,1	244-	U019BAA(U1.QN)/0
93.38	1498	761+80	NO_DETECT	0,1	215+	U0110AAA(U1.Y)/1
93.38	1515	763+80	NO_DETECT	1,15	228+	U016DAA(U1.QN)/1
94.57	1520	763+81	OK	0,2	84+	U013AAA(U4.Y)/1
94.57	1834	844+91	NO_DETECT	21,311	224-	U0126CAA(U1.Q)/0
94.57	1835	845+91	NO_DETECT	0,1	266+	U0111AAA(U1.Y)/1
95.08	1839	845+92	OK	0,2	117-	U0124AAA(U1.Q)/0
95.42	1920	857+93	NO_DETECT	14,42	118+	U0116FAA(U1.Q)/1
95.42	1935	859+94	NO_DETECT	1,13	197-	U014DAA(U1.Q)/0
95.42	1935	1031+94	ABANDONED	2002,20	190-	U0120FAA(U1.Y)/0
95.59	1936	1031+94	OK	0,1	28-	TFLAG/0
95.59	2250	1120+104	NO_DETECT	21,311	65-	U0127CAA(U1.Q)/0
95.59	2267	1122+105	NO_DETECT	1,15	159-	U015DAA(U1.Q)/0
95.76	2330	1133+106	OK	61,42	209+	U0112FAA(U1.QN)/1
95.93	2367	1137+107	OK	1,25	235-	U0145EAA(U1.Y)/0

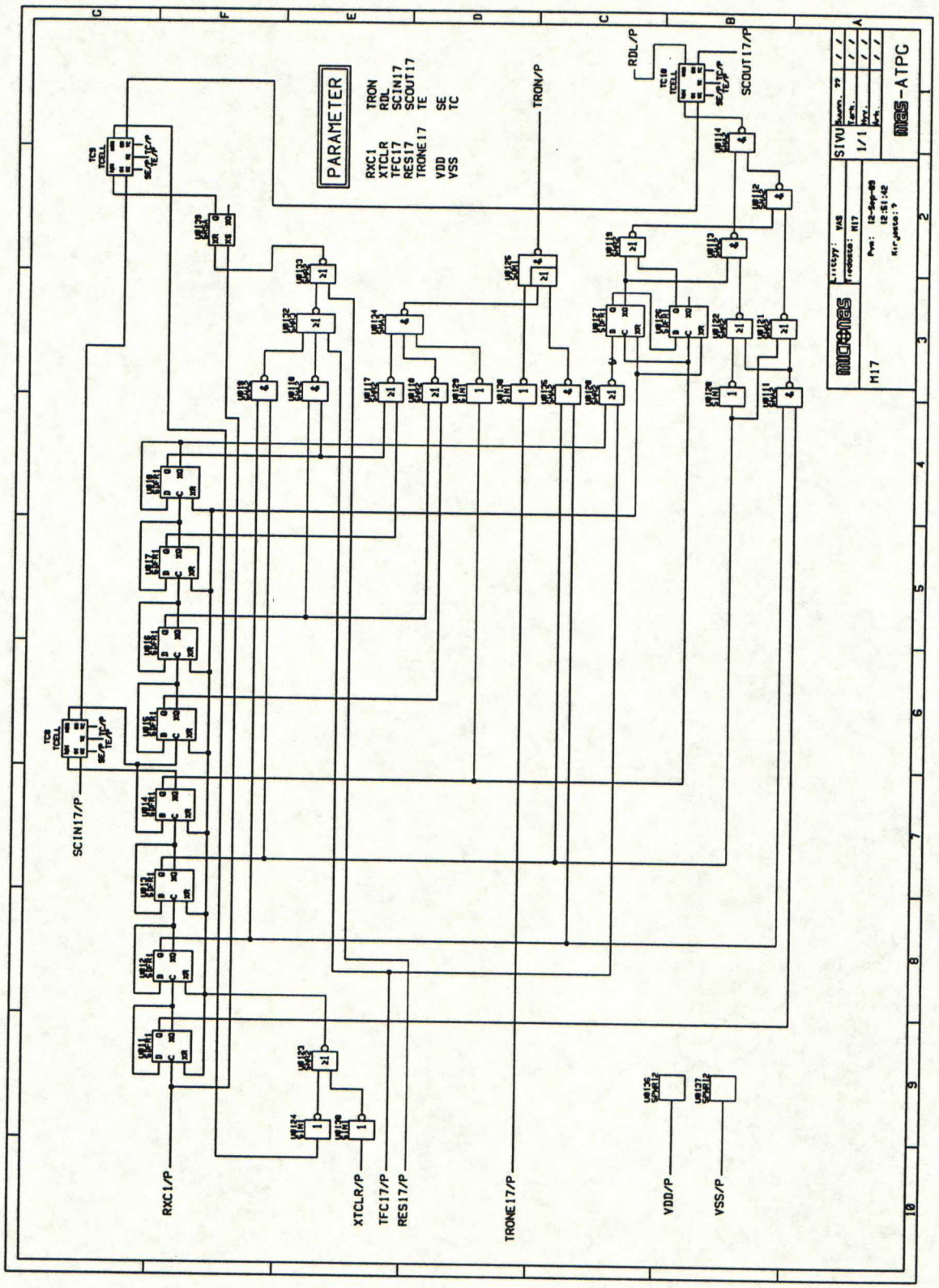

TRON/P
SCOUT/P
FFSKC/P
DIN1/P
M15
M17
M14
M13
M18
M11
TRONE17/M
TFC17/M
RESIN/P
Q2/P
CRFLAG/M
SHM/M
CHM/M
DIN2/P
Q4/P
FSKTUN/M
QS1/P
Q10/P
VDD/P
XTCLR/P
DI11/P
QSCL/M
VSS/P
SCIN/P
SE/P
TC/P
XNOUT/M
MICRONAS
VAS
Pvm : 12-Sep-88
12:41:36
SIVU
1/2
mas-ATPC

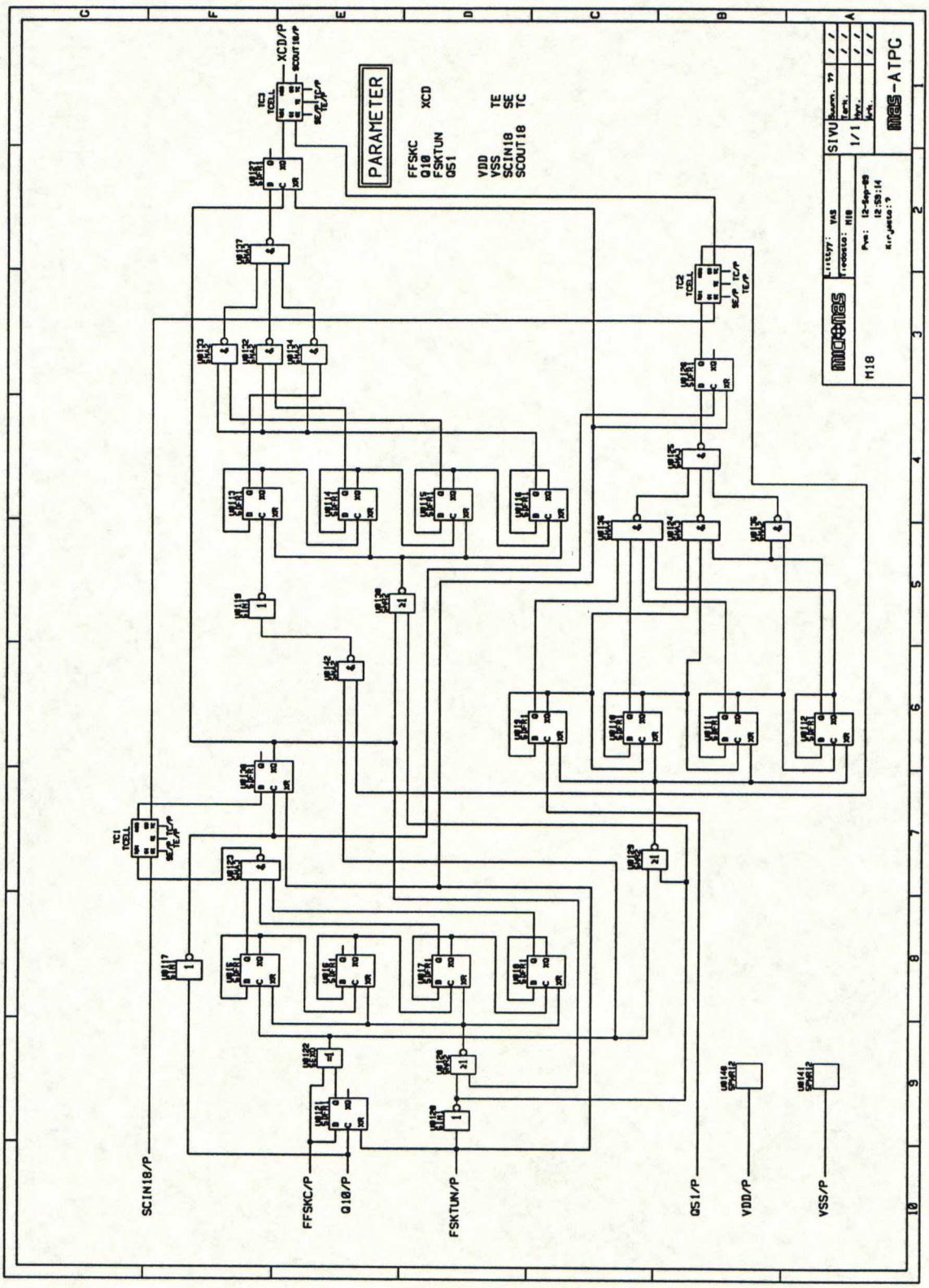

DIN1/P
FFSKC/P
DI11/P
XTCLR/P
QSCL/P
VDD/P
VSS/P
PARAMETER
DIN1
FFSKC
DI11
QSCL
XTCLR
VDD
VSS
SIVU
M11
1/1
-ATPC

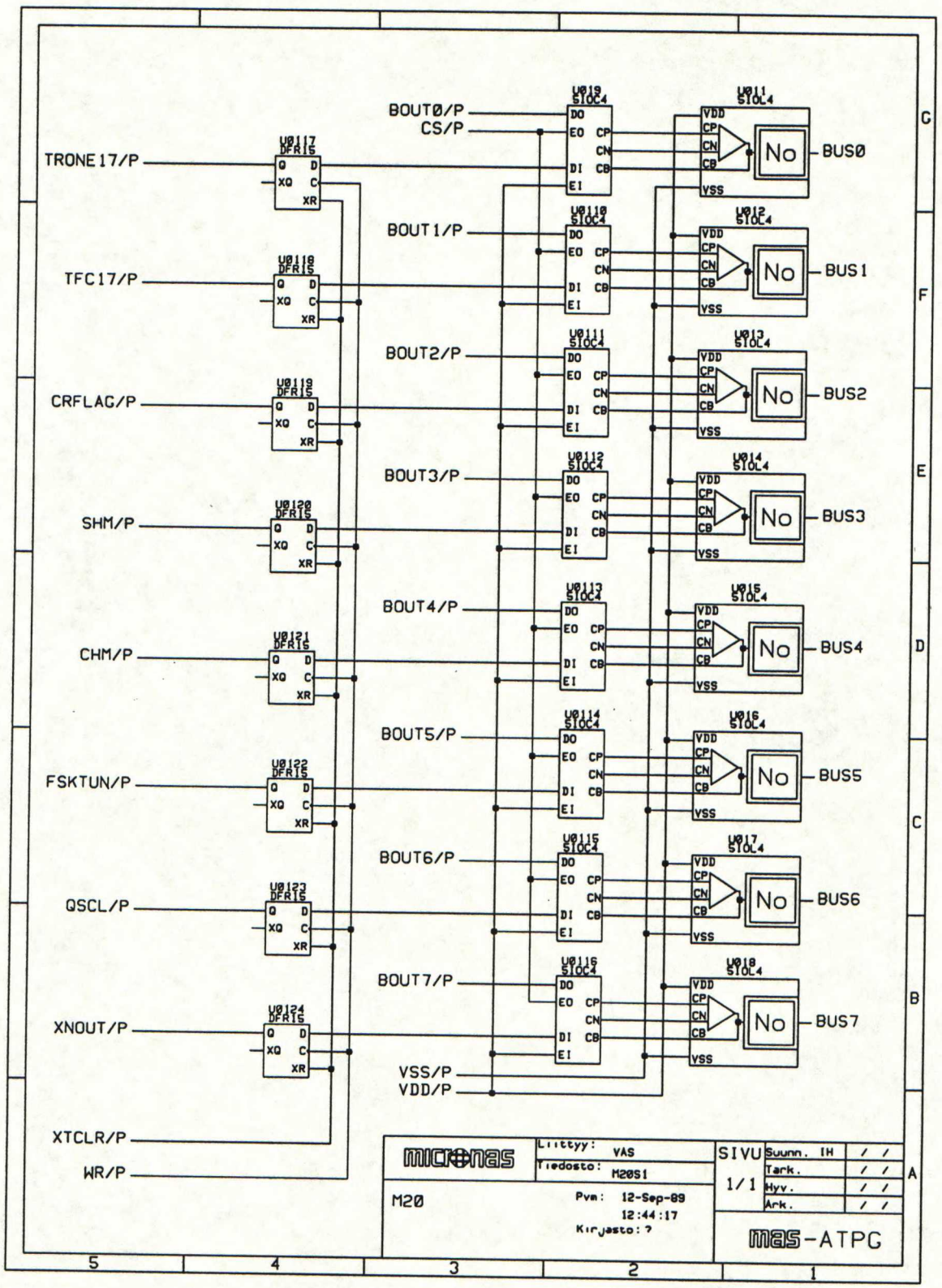

PARAMETER
SCIN13
SCOUT13
Q4
TE
SE
DIN2
TC
XHM
DIN3
DIFF
RXC1
XTCLR
VDD
VSS
RXC1/P
DIFF/P
DIN3/P
Q4/P
DIN2/P
XHM/P
XTCLR/P
VDD/P
VSS/P
SCIN13/P
SCOUT13/P
M13
SIVU
1/1
mas-ATPG


PARAMETER
RXC1
RESIN
DIN3
VDD
VSS
Q2
XTCLR
RDL
CRFLAG
SHM
CHM
SCIN14
SCOUT14
TE
SE
TC
XOUTB0
XOUTB1
XOUTB2
XOUTB3
XOUTB4
XOUTB5
XOUTB6
XOUTB7
RLEN
RFLAG
HM
XHM
RES17
RXC1/P
RESIN/P
DIN3/P
VDD/P
VSS/P
Q2/P
XTCLR/P
RDL/P
VDD/P
CRFLAG/P
SHM/P
CHM/P
XOUTB7/P
XOUTB6/P
XOUTB5/P
XOUTB4/P
XOUTB3/P
XOUTB2/P
XOUTB1/P
XOUTB0/P
RLEN/P
RFLAG/P
HM/P
XHM/P
RES17/P
mas-ATPC
M14
SIVU


CXD1IN/P
CXD0IN/P
XTCLR/P
RLEN/P
XOUTB0/P
RFLAG/P
DIFF/P
BOUT0/P
XOUTB1/P
HM/P
BOUT1/P
XOUTB2/P
TFLAG/P
L3/P
BOUT2/P
XOUTB3/P
TXE/P
L2/P
BOUT3/P
XOUTB4/P
XCD/P
L1/P
RES17/P
BOUT4/P
XOUTB5/P
CFLAG/P
BOUT5/P
XOUTB6/P
AON/P
XNOUT/P
BOUT6/P
XOUTB7/P
BOUT7/P
VDD/P
VSS/P
PARAMETER
CXD1IN L1 BOUT0
CXD0IN RES17 BOUT1
RLEN XOUTB5 BOUT2
XTCLR CFLAG BOUT3
XOUTB0 BOUT4
RFLAG XOUTB6 BOUT5
AON BOUT6
DIFF XNOUT BOUT7
XOUTB1 XOUTB7
HM
VDD
XOUTB2 VSS
TFLAG
L3
XOUTB3
TXE
L2
XOUTB4
XCD
micronas
MIS
SIVU 1/1
mas-ATPG

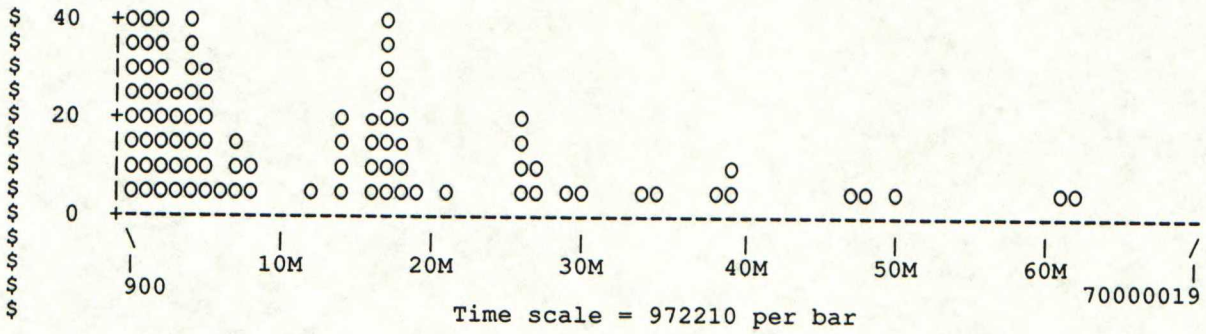

PARAMETER
RXC1
XTCLR
TFC17
RES17
TRONE17
VDD
VSS
TRON
RDL
SCIN17
SCOUT17
TE
SE
TC
SCIN17/P
RXC1/P
XTCLR/P
TFC17/P
RES17/P
TRONE17/P
TRON/P
RDL/P
SCOUT17/P
VDD/P
VSS/P
SIWU
M17
1/1
mas-ATPG


PARAMETER
FFSKC
Q10
FSKTUN
Q51
XCD
VDD
VSS
SCIN18
SCOUT18
TE
SE
TC
SCIN18/P
FFSKC/P
Q10/P
FSKTUN/P
Q51/P
VDD/P
VSS/P
M18
mas-ATPC


TRONE17/P
TFC17/P
CRFLAG/P
SHM/P
CHM/P
FSKTUN/P
QSCL/P
XNOUT/P
XTCLR/P
WR/P
BOUT0/P
CS/P
BOUT1/P
BOUT2/P
BOUT3/P
BOUT4/P
BOUT5/P
BOUT6/P
BOUT7/P
VSS/P
VDD/P
BUS0
BUS1
BUS2
BUS3
BUS4
BUS5
BUS6
BUS7
No
micronas
Liittyy: VAS
Tiedosto: M20S1
M20
Pvm: 12-Sep-89
12:44:17
Kirjasto: ?
SIVU 1/1
Suunn. IH
Tark.
Hyv.
Ark.
mas-ATPG

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  vas.cmd  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                 U N D E T E C T E D   F A U L T S    T A B L E
$
$  S I L O S  I I    ver 1.005         Thu Sep 21 21:29:10 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$ UNDETECTED STUCK-LOW:

.SLOW

+ d15%res17
+ tc4%ndodaa
+ u0114%qfaa
+ u0115%qfaa
+ u0116%qdaa
+ u0117%in2caa
+ u0118%in2caa
+ u0120%qfaa
+ u0130%qdaa
+ u0136%qeaa
+ u0147%in2daa

$ UNDETECTED STUCK-HIGH:

.SHIGH

+ d18%xoutb6
+ te
+ u0113%diaaa
+ u0113%qfaa
+ u0114%diaaa
+ u0114%qfaa
+ u0115%qfaa
+ u0118%qcaa
+ u0120%qcaa
+ u0129%qcaa
+ u0132%qfaa
+ u0132%xqeaa
+ u0133%qfaa
+ u0141%qdaa
+ u0143%inleaa
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  vas.cmd  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$               O S C I L L A T I N G   F A U L T S    T A B L E
$
$  S I L O S  I I    ver 1.005         Thu Sep 21 21:29:10 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$ No table entrys generated.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  vas.cmd  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                    F A U L T   S U M M A R Y    T A B L E
$
$  S I L O S  I I    ver 1.005         Thu Sep 21 21:29:10 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$FIRST STROBE TIME = 900
$LAST  STROBE TIME = 70000000
$STROBE INTERVAL = 1000
$POSSIBLE DETECT THRESHOLD = 5

$FAULT TEST NODES -
$scout
$bus7
$bus6
$bus5
```

```
$bus4
$bus3
$bus2
$bus1
$bus0
$din1
$ffskc
$tron

$                Number              Hard              Possible          Undetected
$                Faulted           Detection           Detection          Faults
$
$              Count     %       Count     %       Count     %       Count     %
$
$.SLOW           283   100.0       272    96.0         0     0.0        11     3.0
$.SHIGH          283   100.0       260    91.0         0     0.0        23     8.0
$.ISLOW            0     0.0         0     0.0         0     0.0         0     0.0
$.ISHIGH           0     0.0         0     0.0         0     0.0         0     0.0
$TOTAL           566   100.0       532    93.0         0     0.0        34     6.0
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  vas.cmd  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$
$$                F A U L T   H I S T O G R A M   H I S T O G R A M$
$$
$$  S I L O S  I I    ver 1.005          Thu Sep 21 21:29:10 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$DETECTIONS AND OSCILLATIONS VS. TIME
$Legend: O=OUTPUT DETECTED I=INPUT DETECTED
$        o=OUTPUT POSSIBLE i=INPUT POSSIBLE @=OSCILLATION
$       |
$       |
$       |
$  220  +
$       |
$       |
$       |
$  200  +
$       |
$       |
$       |
$  180  +
$       |
$       |
$       |
$  160  +
$       |
$       |
$       |
$  140  +
$       |
$       |
$       |
$  120  +
$       |o
$       |o
$       |o
$
$  100  +O
$       |O
$       |O
$       |O
$   80  +O o
$       |O O
$       |O O
$       |O O o
$   60  +O O o
$       |OoO O
$       |OoO O
$       |OOO O
```


40
20
0
900
10M
20M
30M
40M
50M
60M
70000019
Time scale = 972210 per bar

INTELLIGEN FAULT LOG SUMMARY
CIRCUIT "vas"

SUMMARY OF COMPLETION INFORMATION FOR ALL BUILDS ATTEMPTED:

COMPLETION STATUS GROUP	NUMBER OF BUILDS IN THIS GROUP	FRACTION OF TOTAL BUILDS IN THIS GROUP	ACTUAL NUMBER OF FAULTS REPRESENTED IN THIS GROUP
NO_TEST	1	0.0476	1
ABANDONED	11	0.5238	13
ABORTED	2	0.0952	3
TIMEOUT	3	0.1429	5
NO_DETECT	1	0.0476	1
OK	3	0.1429	3
TOTAL BUILDS:	21		

TOTAL UNTESTABLE FAULTS (NO_PATH + NO_EXERCI): 0

STATISTICS FOR TESTS BUILT (OK, NO_DETECT, NO_SOLID):

AVERAGE NUMBER OF VECS: 20
MAXIMUM VECTORS: 30
MAXIMUM BACKUPS: 22

AVERAGE ATG TIME: 4s
MAXIMUM ATG TIME: 6s
STANDARD DEVIATION: 2.24

DETECTION STATUS WHEN THIS REPORT WAS CREATED:

TOTAL NUMBER OF FAULTS SELECTED: 26
TOTAL CLASSES: 21
FAULTS PER PASS: 1500
INDEX IN FAULT LIST OF LAST BUILD: 19
INDEX OF NEXT PASS BOUNDARY: 21
COMPLETION: 1.0000

NUMBER OF FAULTS NOT DETECTED: 22
NUMBER OF OSCILLATION DETECTS: 0
NUMBER OF POTENTIAL DETECTS: 1
NUMBER OF SOLID DETECTS: 3
TOTAL DETECTS: 4

ESTIMATED DETECTION:

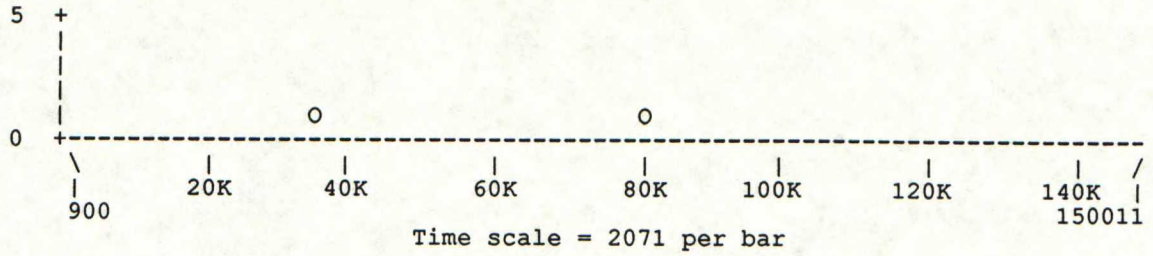
$$\frac{\text{TOTAL DETECTS}}{(\text{TOTAL FAULTS} * \text{COMPLETION}) - \text{UNTESTABLE}} = \%15.38$$

=== INTELLIGEN FAULT LOG ===
CIRCUIT: "vas"
Fri Sep 22 10:04:53 1989
21 ENTRIES

DETECT	CYC	A+S	STATUS	B,V	ID	FAULT NAME
0.00	1	8+2	ABANDONED	201,2	184+3099.2	U0113FAA(U1.D)/1
0.00	1	25+2	ABANDONED	201,6	95+	U0132EAA(U1.QN)/1
0.00	1	32+2	ABORTED	4,13	255+	U0141DAA(U1.Y)/1
0.00	1	90+2	ABANDONED	201,31	194+	U013EAA(U1.QN)/1
0.00	1	330+2	TIMEOUT	39,561	170+	U0132FAA(U1.Y)/1
0.00	1	330+2	NO_TEST	2,1	94+	TCTRL1(TE.Y)/1
0.00	1	338+2	ABORTED	3,13	251-	U0153DAA(U1.Q)/0
0.00	1	562+2	ABANDONED	201,176	296-	U0143EAA(U1.Y)/0
0.00	1	581+2	ABANDONED	210,8	279+	U0118CAA(U1.Y)/1
0.00	1	599+2	ABANDONED	218,8	241-	U017CAA(U1.Q)/0
0.00	1	839+2	TIMEOUT	85,544	77+3677.2	U0115FAA(U1.D)/1
0.00	1	847+2	ABANDONED	201,2	131+3387.2	U0114FAA(U1.D)/1
0.00	1	882+2	ABANDONED	201,23	180+	U0120CAA(U1.Y)/1
0.00	1	901+2	ABANDONED	210,8	239-	U015CAA(U1.Q)/0
0.00	1	919+2	ABANDONED	201,11	294+	U0129CAA(U1.Y)/1
0.00	1	1159+2	TIMEOUT	39,560	102+	U0133FAA(U1.Y)/1
3.84	6	1160+2	OK	0,3	50+	U0113AAA(U4.Y)/1
11.53	38	1163+3	OK	0,27	127-	TC4DAA(NDO.Y)/0
15.38	81	1169+4	OK	1,30	103+	U0114AAA(U4.Y)/1
15.38	113	1174+5	NO_DETECT	22,22	199-	U0120FAA(U1.Y)/0
15.38	113	1209+5	ABANDONED	201,43	99-	U0116DAA(U1.Q)/0

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  VAS.CMD  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                  U N D E T E C T E D   F A U L T S    T A B L E
$
$  S I L O S  I I    ver 1.005        Fri Sep 22 10:22:35 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$ UNDETECTED STUCK-LOW:

.SLOW

+ d15%res17
+ u0114%qfaa
+ u0115%qfaa
+ u0116%qdaa
+ u0117%in2caa
+ u0118%in2caa
+ u0120%qfaa
+ u0130%qdaa
+ u0136%qeaa
+ u0147%in2daa

$ UNDETECTED STUCK-HIGH:

.SHIGH

+ d18%xoutb6
+ te
+ u0113%diaaa
+ u0113%qfaa
+ u0114%qfaa
+ u0115%qfaa
+ u0118%qcaa
+ u0120%qcaa
+ u0129%qcaa
+ u0132%qfaa
+ u0132%xqeaa
+ u0133%qfaa
+ u0141%qdaa
+ u0143%in1eaa
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  VAS.CMD  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                O S C I L L A T I N G   F A U L T S    T A B L E
$
$  S I L O S  I I    ver 1.005        Fri Sep 22 10:22:35 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$ No table entrys generated.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  VAS.CMD  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                     F A U L T   S U M M A R Y    T A B L E
$
$  S I L O S  I I    ver 1.005        Fri Sep 22 10:22:35 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
FIRST STROBE TIME = 900
LAST  STROBE TIME = 150000
STROBE INTERVAL = 1000
POSSIBLE DETECT THRESHOLD = 5

FAULT TEST NODES -
scout
bus7
bus6
bus5
bus4
bus3
```

```
bus2
bus1
bus0
din1
ffskc
tron

                Number                  Hard                Possible              Undetected
                Faulted                 Detection           Detection             Faults

               Count     %           Count     %           Count     %           Count     %

.SLOW             11   100.0            1     9.0            0     0.0           10    90.0
.SHIGH            15   100.0            1     6.0            0     0.0           14    93.0
.ISLOW             0     0.0            0     0.0            0     0.0            0     0.0
.ISHIGH            0     0.0            0     0.0            0     0.0            0     0.0
TOTAL             26   100.0            2     7.0            0     0.0           24    92.0
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$  VAS.CMD  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$                F A U L T   H I S T O G R A M   H I S T O G R A M
$
$  S I L O S  I I    ver 1.005          Fri Sep 22 10:22:35 1989
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
DETECTIONS AND OSCILLATIONS VS. TIME
Legend: O=OUTPUT DETECTED I=INPUT DETECTED
        o=OUTPUT POSSIBLE i=INPUT POSSIBLE @=OSCILLATION
      |
      |
   45 +
      |
      |
      |
      |
   40 +
      |
      |
      |
      |
   35 +
      |
      |
      |
      |
   30 +
      |
      |
      |
      |
   25 +
      |
      |
      |
      |
   20 +
      |
      |
      |
      |
   15 +
      |
      |
      |
      |
   10 +
      |
      |
      |
      |
```



Time scale = 2071 per bar